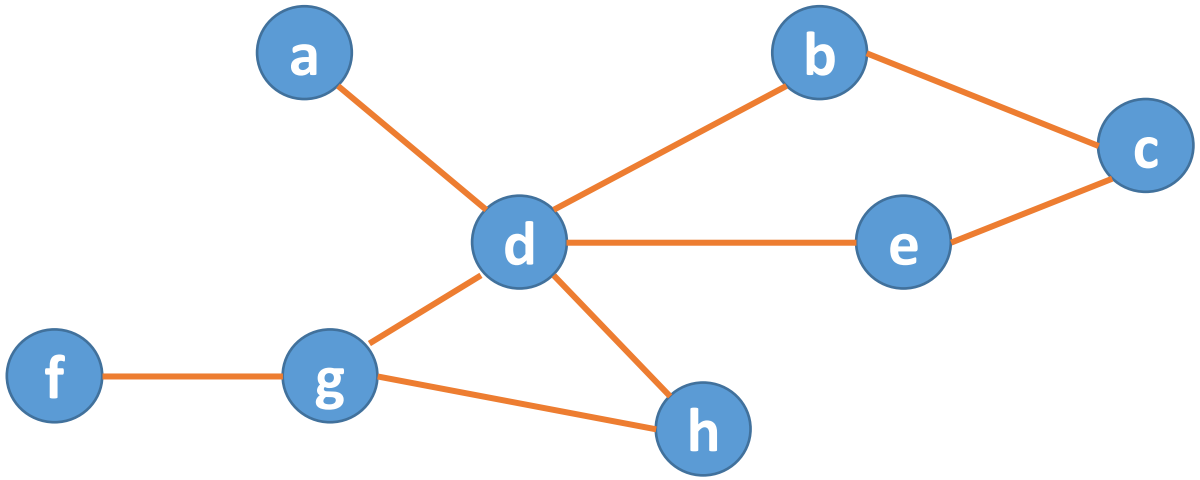
Chapitre VI

Théorie des graphes

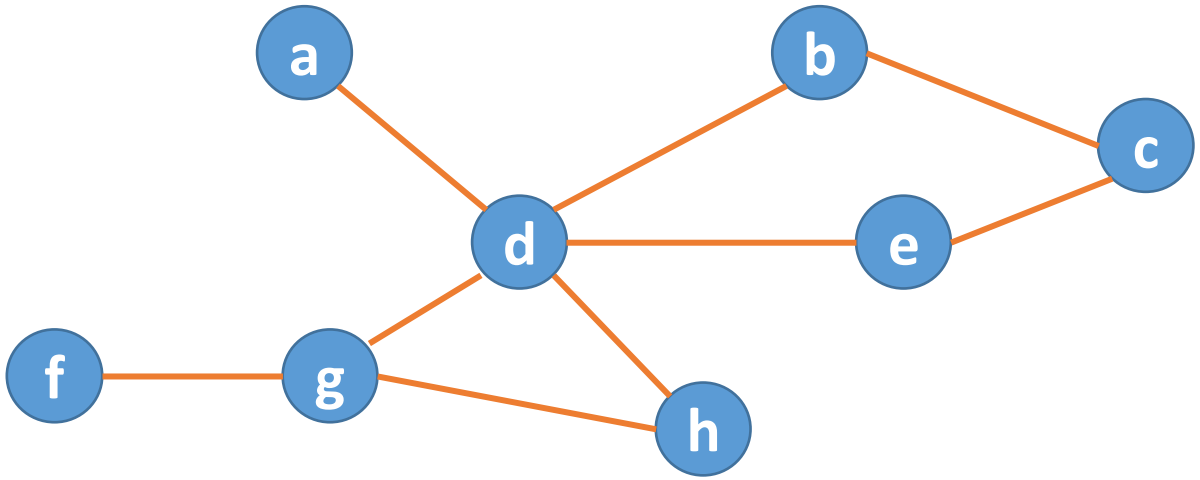
Définition

Un graphe permet de décrire un ensemble d'objets et leurs relations, c'est-à-dire les liens entre les objets



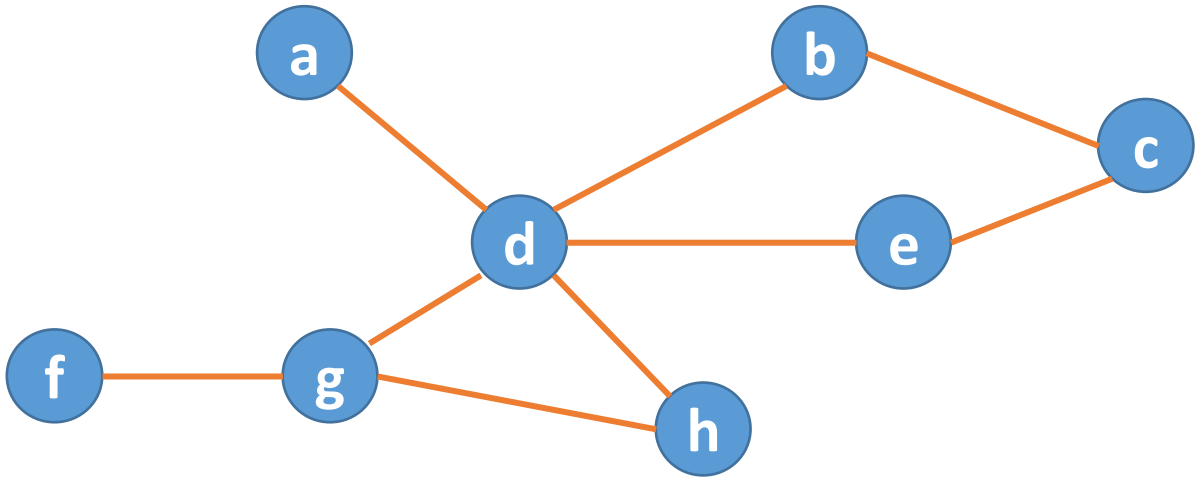
Définition

Les objets sont appelés les **nœuds** ou les **sommets** du graphe. Ils sont regroupés dans l'ensemble **X={a, b, c, d, e, f, g, h}**



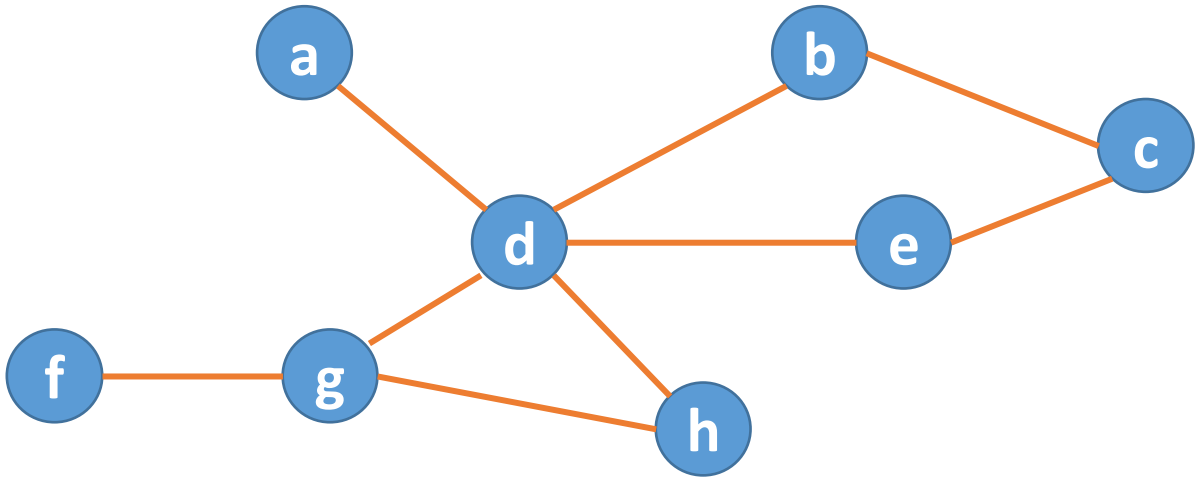
Définition

Un lien entre deux objets est appelé une **arête** ou un **arc**. Une arête identifie une paire de sommets (x_i, x_j) qui sont reliés entre eux. U représente l'ensemble des arêtes: **U={(a,d), (d,b), (b,c),…, (f,g), (g,h)}**



Définition

On note le **graphe non orienté**: **G(X,U)**



Utilisation

Les graphes modélisent de nombreuses situations concrètes où interviennent des objets en interaction

- Les interconnexions routière, ferroviaire ou aérienne entre différentes agglomérations
- Les liens entre les composants d'un circuit électronique
- Le plan d'une ville et de ses rues
- …

Utilisation

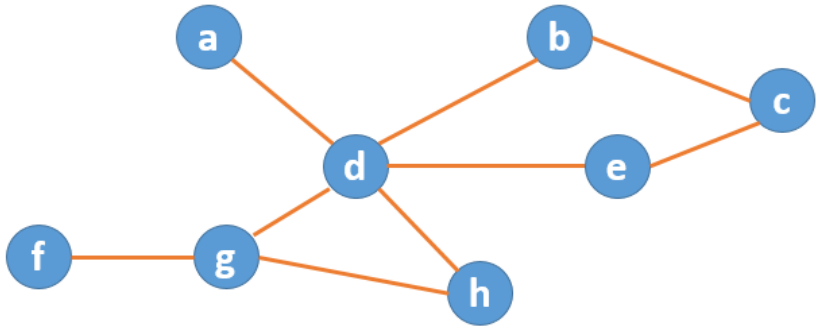
L’ensemble des techniques et outils mathématiques mis au point dans la théorie des graphes permettent de déduire des méthodes de résolutions et des algorithmes pour résoudre des problèmes tels que

- Quel est le plus court chemin pour se rendre d’une ville à une autre?
- Comment minimiser la longueur totale des connexions d’un circuit?
- Peut-on mettre une rue à sens unique sans rendre impossible la circulation en ville?

Définition

Un graphe est **simple** si au plus une arête (0 ou 1) relie deux sommets et s'il n'y a pas de boucle (un sommet qui est relié à lui-même) sur un sommet

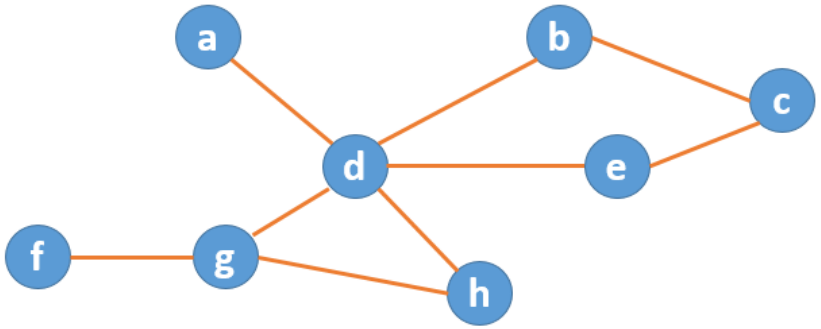
Est-ce que le graphe suivant est simple?



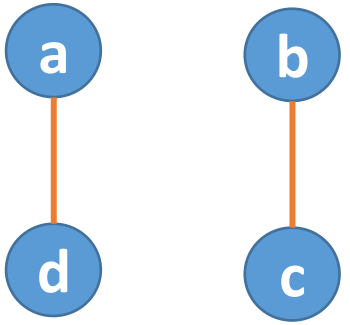
Définition

Un graphe est **connexe** s'il est possible, à partir de n'importe quel sommet, de rejoindre tous les autres en suivant les arêtes.

Connexe

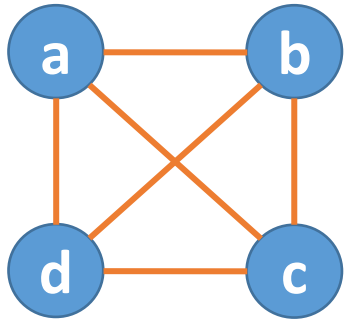


Non connexe



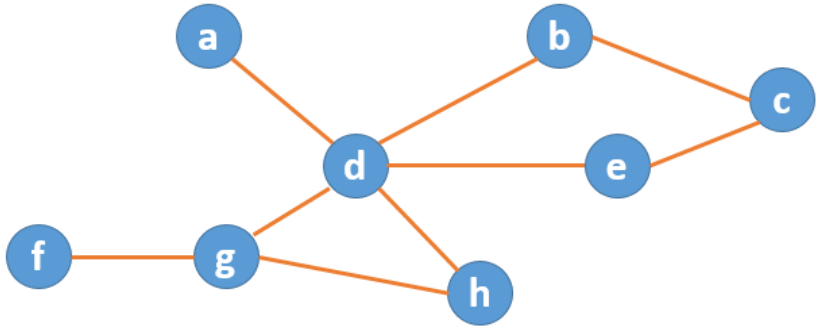
Définition

Un graphe est **complet** si chaque sommet du graphe est relié directement à tous les autres sommets du graphe



Définition

On appelle **degré** d'un sommet x, et on note **d(x)**, le nombre d'arêtes incidentes à ce sommet.



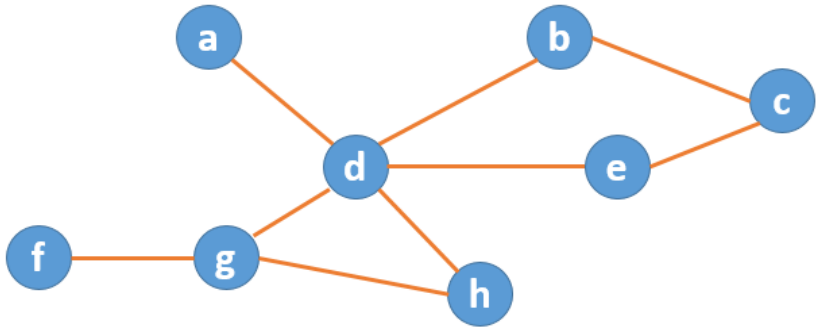
Sommet	Degré
A	1
B	2
C	2
D	5

Sommet	Degré
E	2
F	1
G	3
H	2

Théorème (*Lemme des poignées de main*): La somme des degrés des sommets d'un graphe est égale à deux fois le nombre d'arêtes

Définition

Le **degré d'un graphe** est le degré maximum de tous ses sommets.

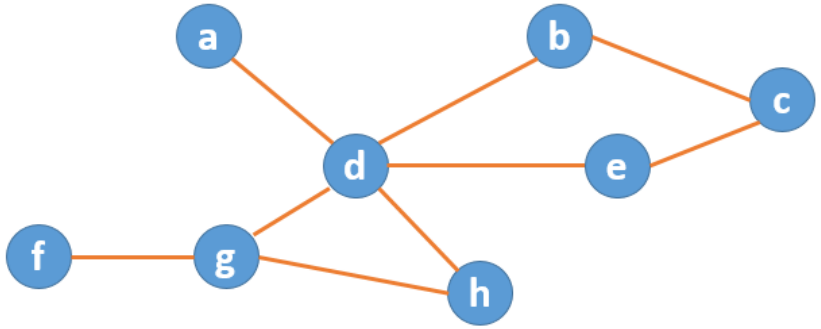


Degré du graphe = 5

Un graphe dont tous les sommets ont le même degré est dit **régulier**

Définition

Une chaîne dans un graphe est une suite ayant pour éléments alternativement des sommets et des arêtes, commençant par un sommet et se terminant par un sommet, et telle que chaque arête est encadrée par ses extrémités



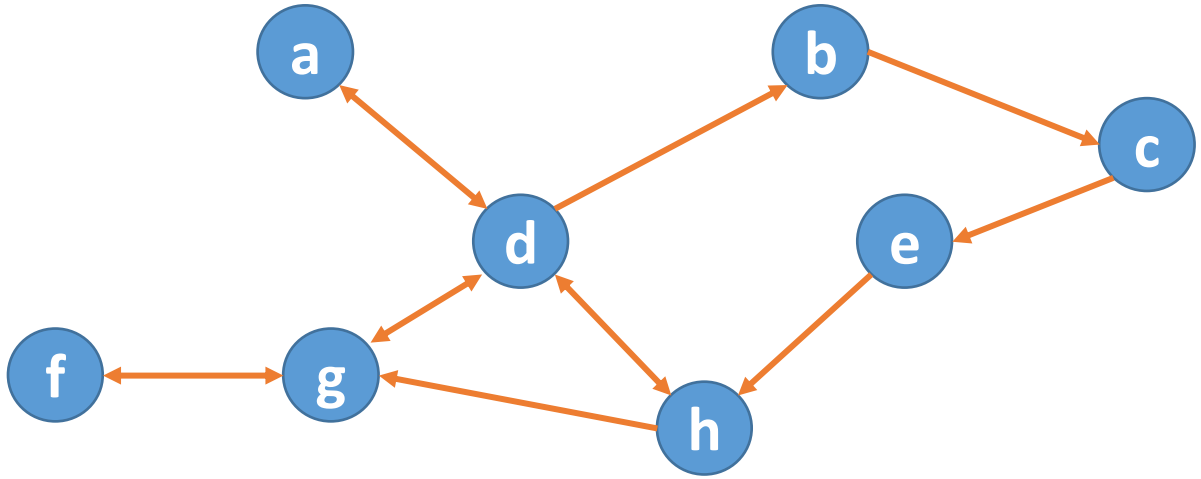
a – (a,d) – d – (d,h) – h – (h,g) – g (longueur : 3)

La longueur d’une chaîne est le nombre d’arêtes de la chaîne.

Définition

Un graphe orienté est un graphe dont les arêtes ont un sens.

Un arc est une arête orientée qui est défini par une extrémité initiale et une extrémité finale



Définition

On note $d^+(x)$ le degré extérieur d'un sommet, c'est-à-dire le nombre d'arcs ayant x comme extrémité initiale

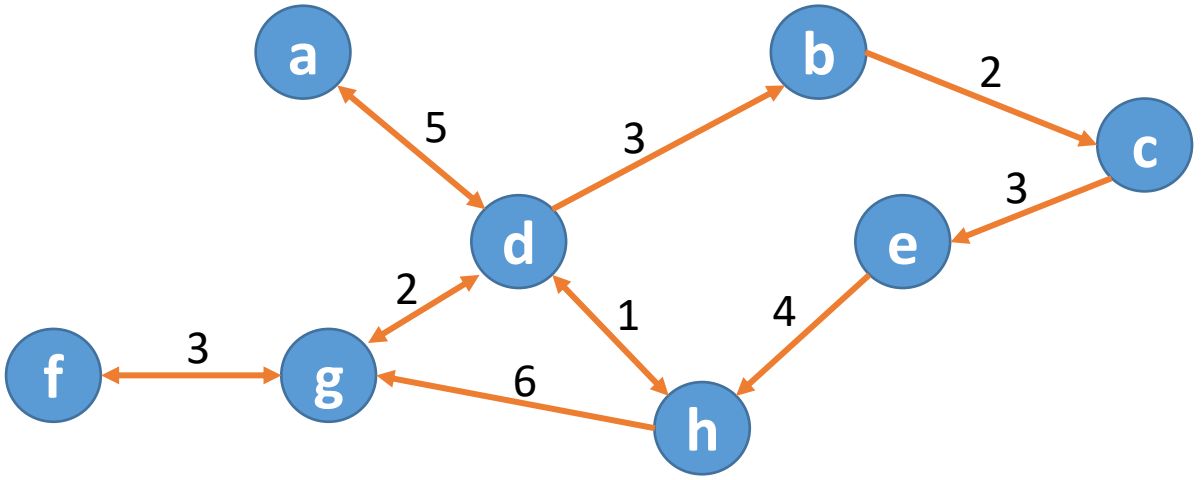
On note $d^-(x)$ le degré intérieur d'un sommet, c'est-à-dire le nombre d'arcs ayant x comme extrémité finale

$$d(x) = d^+(x) + d^-(x)$$

Définition

Un chemin correspond à une chaîne lorsque le graphe est orienté

La longueur d'un chemin est la somme des longueurs de chacun des arcs qui le composent



Définition

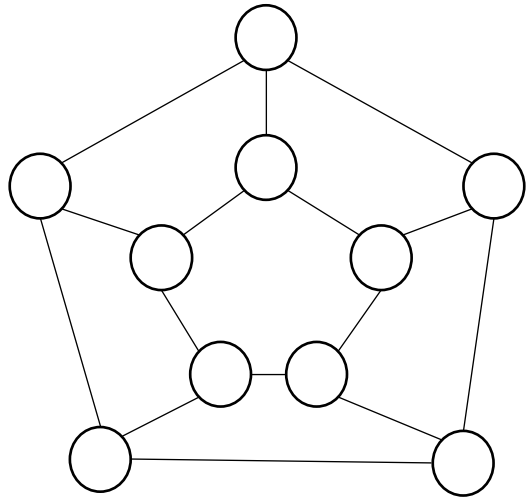
Un graphe valué peut être utilisé pour représenter

- Les coûts de transit entre chaque sommet
- La distance en km entre chaque sommet
- Le temps nécessaire pour parcourir les arêtes
- …

Coloriage de graphe – Le jeu de Snort

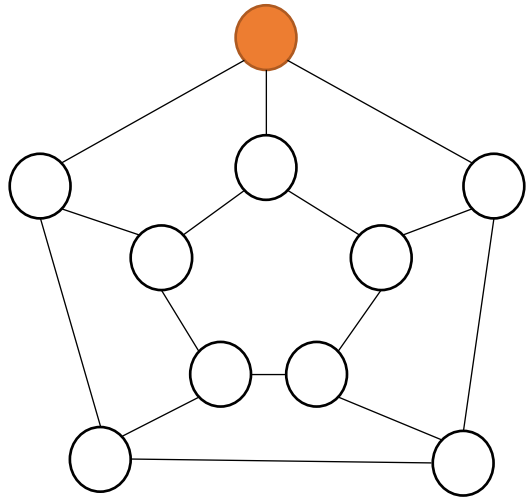
Le jeu se joue à 2 à partir d'un graphe non orienté et connexe.

Par exemple, considérons le graphe suivant:



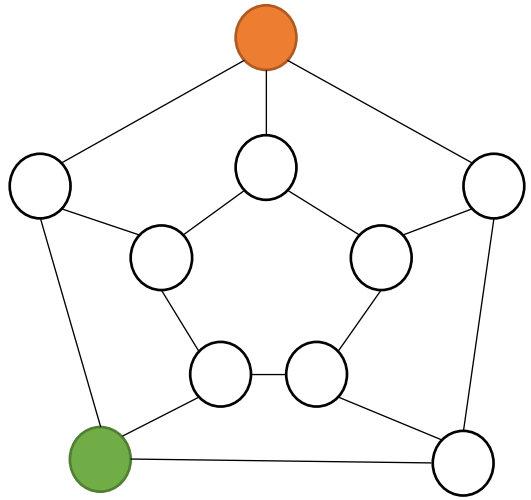
Coloriage de graphe – Le jeu de Snort

Le premier joueur choisit un nœud et le colorie dans sa couleur



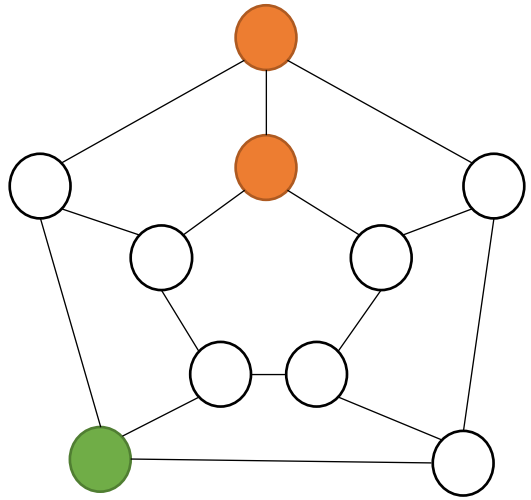
Coloriage de graphe – Le jeu de Snort

Le deuxième joueur choisit un nœud de telle sorte qu'il ne colorie pas un nœud adjacent à celui du premier joueur



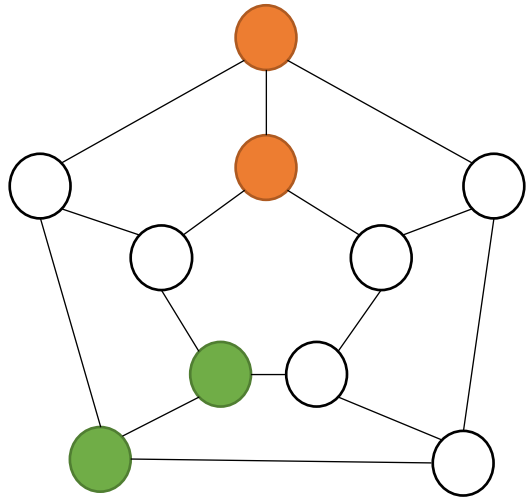
Coloriage de graphe – Le jeu de Snort

…



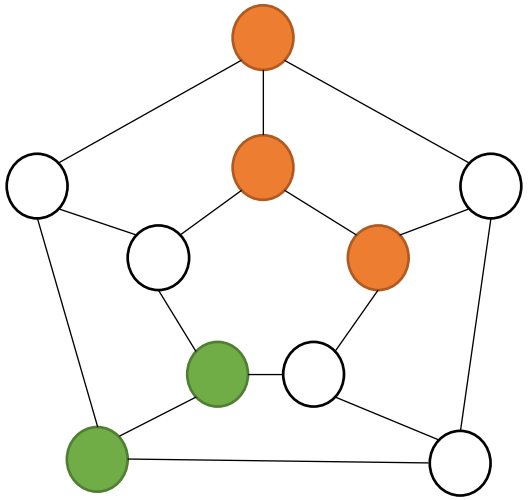
Coloriage de graphe – Le jeu de Snort

…



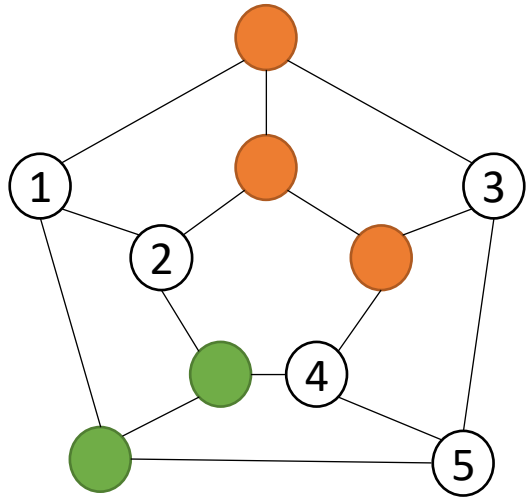
Coloriage de graphe – Le jeu de Snort

…



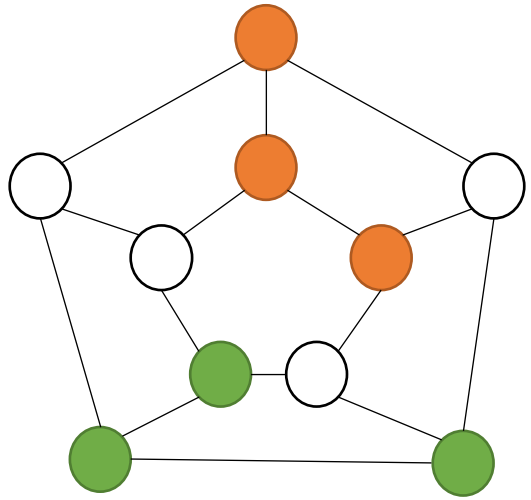
Coloriage de graphe – Le jeu de Snort

Il reste alors un seul choix pour le deuxième joueur.



Coloriage de graphe – Le jeu de Snort

Et le premier joueur perd …



Coloriage de graphe – Le jeu de Snort

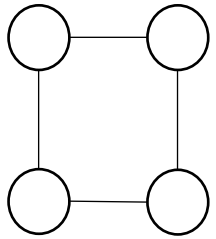
L’intérêt du jeu est relatif.

Mais peut-on trouver un algorithme qui permet de prédire le gagnant?

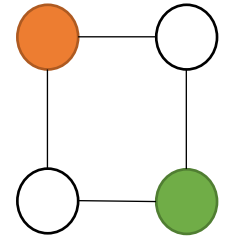
Coloriage de graphe – Le jeu de Snort

Ce qu'on sait pour l'instant:

- Pour un graphe complet, c'est le premier joueur qui gagne
- Pour un graphe incomplet simple, tel que



C'est le deuxième joueur qui gagne. Le premier joue et le second doit simplement colorier le nœud opposé



Coloriage de graphe – Le jeu de Snort

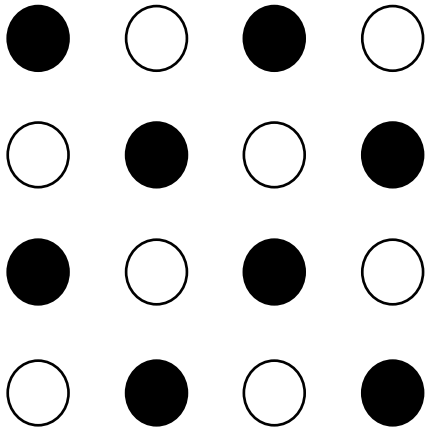
Ce qu’on sait pour l’instant:

- Pour des graphes plus compliqués, il est plus difficile de déterminer la stratégie gagnante et dans la plupart des cas, on ne connait pas encore de stratégie gagnante

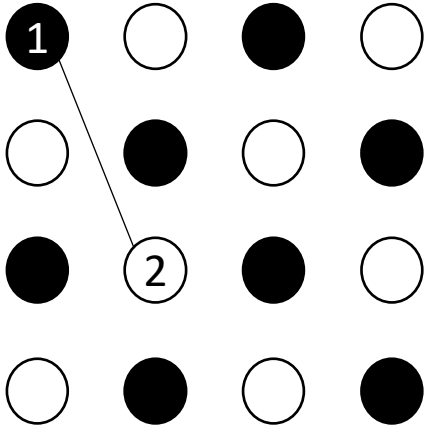
Le parcours du cavalier

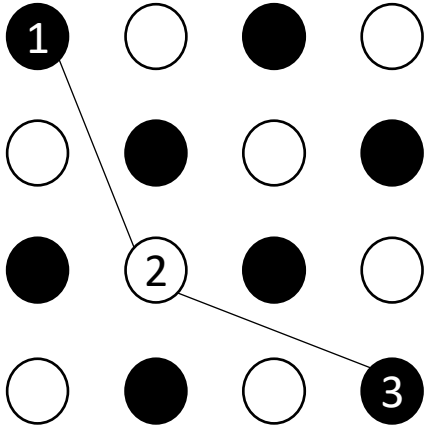
L’objectif est de faire parcourir à un cavalier chaque case de l’échiquier une après l’autre.

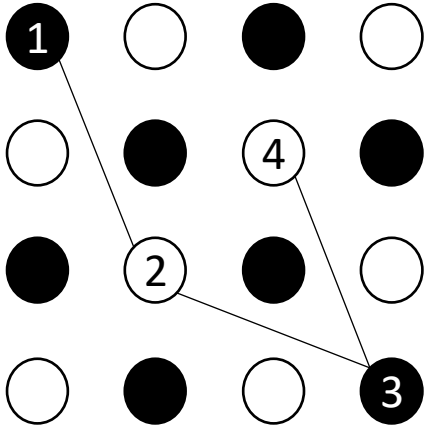
Euler a d’abord étudié le problème à partir d’un échiquier 4x4

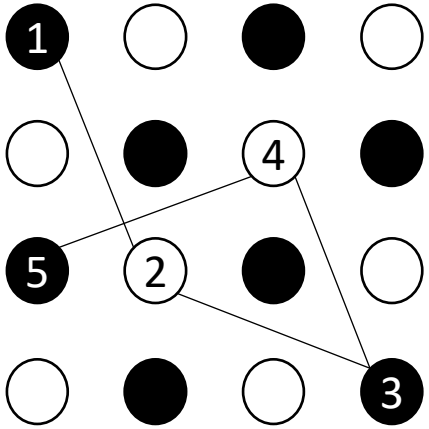


Le parcours du cavalier - Euler

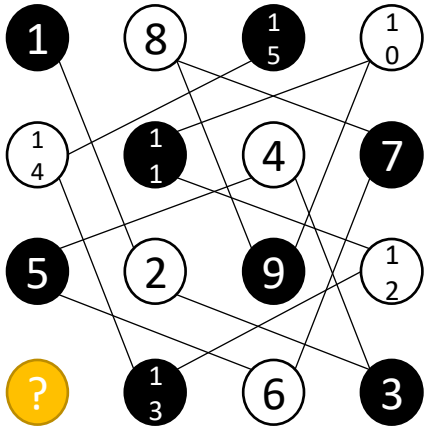








Euler a démontré qu'il n'existe aucun chemin pour l'échiquier 4x4 permettant au cavalier de parcourir toutes les cases.



Le parcours du cavalier

A l’heure actuelle, on sait qu’il existe

- 26 534 728 821 064 circuits fermés différents sur un échiquier 8x8, pour 19 591 828 170 979 904 circuits ouverts
- 9862 circuits fermés sur un échiquier 6x6, pour 6 637 920 circuits ouverts

Application – Le chemin le plus court

Lorsqu'un chemin existe entre deux sommets dans un graphe, on peut se demander si ce chemin est le **plus court** possible.

Lorsque le graphe n'est pas valué, on cherche simplement à trouver le chemin qui contient le moins d'arêtes

Lorsque le graphe est valué, la question n'est plus simplement de trouver le chemin le plus court, mais bien celle de trouver le chemin dont le poids des arêtes sera le plus faible possible

L'objectif est donc de trouver un algorithme qui nous permette de trouver le chemin correspondant au poids minimum

Application – Le chemin le plus court

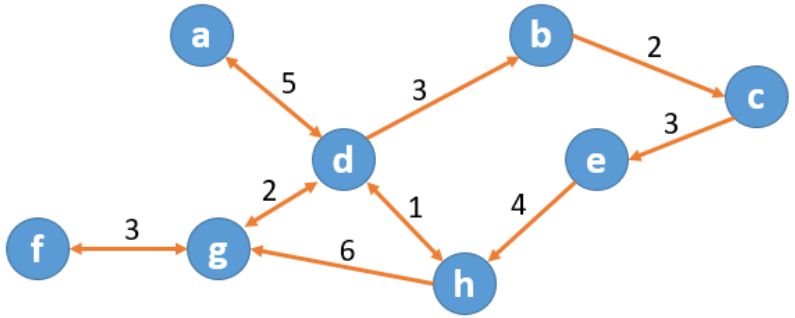
Un graphe valué peut être représenté par une matrice carrée, dont les coefficients correspondent à la valuation des arcs entre chaque sommet.

Soit un graphe valué dont on a numéroté les sommets de 1 à n

La matrice carrée $M = m_{ij}$ d'ordre n est définie par

$$m_{ij} = \begin{cases} v(i,j) si (i,j) \in U \\ +\infty \; sinon \end{cases}$$

Application – Le plus court chemin



$$\begin{pmatrix} \infty & \infty & \infty & \mathbf{5} & \infty & \infty & \infty & \infty \\ \infty & \infty & \mathbf{2} & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \mathbf{3} & \infty & \infty & \infty \\ \mathbf{5} & \mathbf{3} & \infty & \infty & \mathbf{2} & \infty & \infty & \mathbf{1} \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \mathbf{4} \\ \infty & \infty & \infty & \infty & \infty & \infty & \mathbf{3} & \infty \\ \infty & \infty & \infty & \mathbf{2} & \infty & \mathbf{3} & \infty & \infty \\ \infty & \infty & \infty & \mathbf{1} & \infty & \infty & \mathbf{6} & \infty \end{pmatrix}$$

Algorithme de Bellman

Trouver le chemin le plus court à partir d'un sommet donné et d'un graphe comprenant k sommets.

On va calculer itérativement une matrice de k colonnes en ajoutant 1 ligne à chaque itération

La matrice est initialisée comme suit:

- 0 pour le sommet de départ
- ∞ pour les autres sommets

Algorithme de Bellman

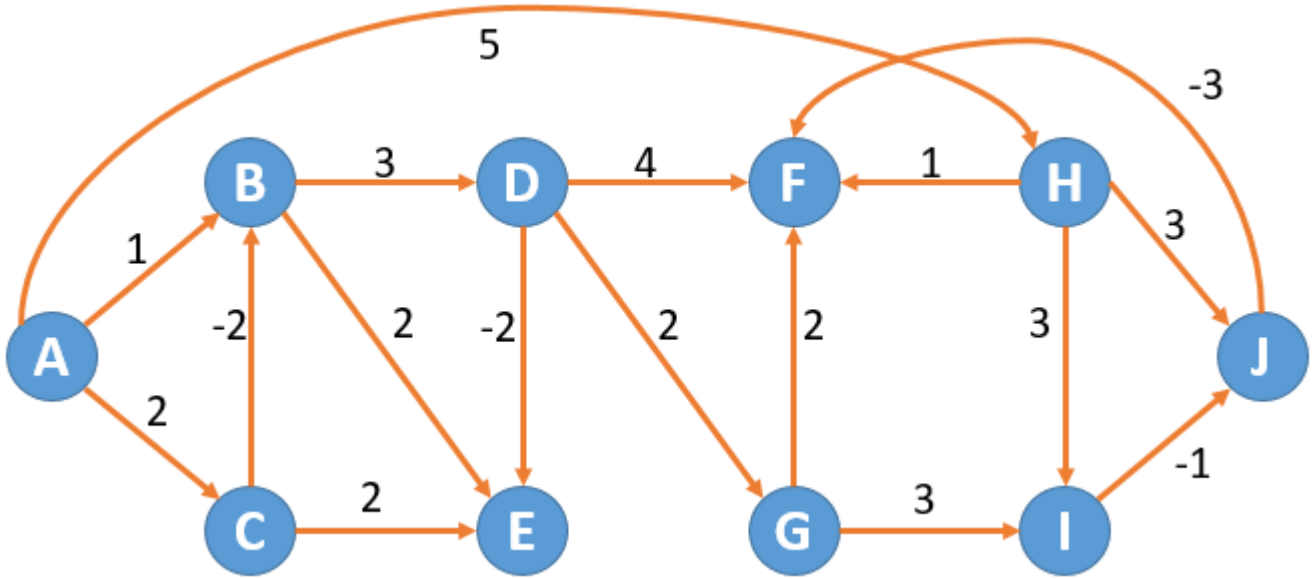
A chaque itération, on part des sommets atteints et on se déplace vers les sommets atteignables.

On compare la longueur du chemin déjà obtenu pour un sommet atteignable avec celle obtenue du sommet origine et de la distance entre les 2 sommets. On garde la valeur minimale

On arrête d'itérer lorsque soit

- On vient de terminer l'itération k-1
- Les distances trouvées sont identiques aux distances de l'itération précédente

Algorithme de Bellman



	A	B	C	D	E	F	G	H	I	J
A	∞	1	2	∞	∞	∞	∞	5	∞	∞
B	∞	∞	∞	3	2	∞	∞	∞	∞	∞
C	∞	-2	∞	∞	2	∞	∞	∞	∞	∞
D	∞	∞	∞	∞	-2	4	2	∞	∞	∞
E	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
F	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
G	∞	∞	∞	∞	∞	2	∞	∞	3	∞
H	∞	∞	∞	∞	∞	1	∞	∞	3	3
I	∞	∞	∞	∞	∞	∞	∞	∞	∞	-1
J	∞	∞	∞	∞	∞	-3	∞	∞	∞	∞

Algorithme de Bellman

Sommet de départ: A

Trouver le plus court chemin pour atteindre les autres sommets

Initialisation

k	A		B		C		D		E		F		G		H		I		J	
0	0		∞		∞		∞		∞		∞		∞		∞		∞		∞	

Algorithme de Bellman

Occurrence 1

A partir de A, on peut atteindre B (1), C (2) et H (5)

k	A		B		C		D		E		F		G		H		I		J	
0	0		∞		∞		∞		∞		∞		∞		∞		∞		∞	
1			1	A	2	A	∞		∞		∞		∞		5	A	∞		∞	

Distance parcourue

Sommet précédent

Algorithme de Bellman

Occurrence 2

On repart des sommets atteints à partir de l'occurrence 1

- De B, on peut atteindre D (1 + 3) et E (1 + 2)
- De C, on peut atteindre B (2 – 2) et E (2 + 2)
- De H, on peut atteindre F (5 + 1), I (5 + 3) et J (5 + 3)

k	A		B		C		D		E		F		G		H		I		J	
0	0		∞		∞		∞		∞		∞		∞		∞		∞		∞	
1			1	A	2	A	∞		∞		∞		∞		5	A	∞		∞	
2			0	C			4	B	3	B	6	H	∞				8	H	8	H

Algorithme de Bellman

Occurrence 3

On repart des sommets atteints à partir de l'occurrence 2

- De B, on peut atteindre D (0 + 3) et E (0 + 2)
- De D, on peut atteindre E (4 – 2), F (4 + 4) et G (4 + 2)
- De E, on ne peut atteindre aucun sommet
- De F, on ne peut atteindre aucun sommet
- De I, on peut atteindre J (8 – 1)
- De J, on peut atteindre F (8 – 3)

k	A		B		C		D		E		F		G		H		I		J	
0	0		∞		∞		∞		∞		∞		∞		∞		∞		∞	
1			1	A	2	A	∞		∞		∞		∞		5	A	∞		∞	
2			0	C			4	B	3	B	6	H	∞				8	H	8	H
3							3	B	2	B	5	J	6	H					7	I

Algorithme de Bellman

Occurrence 4

On repart des sommets atteints à partir de l'occurrence 3

- De D, on peut atteindre E (3 – 2), F (3 + 4) et G (3 + 2)
- De E, on ne peut atteindre aucun sommet
- De F, on ne peut atteindre aucun sommet
- De G, on peut atteindre F (6 +2) et I (6 + 3)
- De J, on peut atteindre F (7 – 3)

k	A		B		C		D		E		F		G		H		I		J	
0	0		∞		∞		∞		∞		∞		∞		∞		∞		∞	
1			1	A	2	A	∞		∞		∞		∞		5	A	∞		∞	
2			0	C			4	B	3	B	6	H	∞				8	H	8	H
3							3	B	2	B	5	J	6	H					7	I
4									1	D	4	J	5	D			8	H		

Algorithme de Bellman

Occurrence 5

On repart des sommets atteints à partir de l'occurrence 4

- De E, on ne peut atteindre aucun sommet
- De F, on ne peut atteindre aucun sommet
- De G, on peut atteindre F (6 +2) et I (6 + 3)
- De I, on peut atteindre J (8 – 1)

k	A		B		C		D		E		F		G		H		I		J	
0	0		∞		∞		∞		∞		∞		∞		∞		∞		∞	
1			1	A	2	A	∞		∞		∞		∞		5	A	∞		∞	
2			0	C			4	B	3	B	6	H	∞				8	H	8	H
3							3	B	2	B	5	J	6	H					7	I
4									1	D	4	J	5	D			8	H		
5											4	J					8	H	7	I

Algorithme de Bellman

Occurrence 6

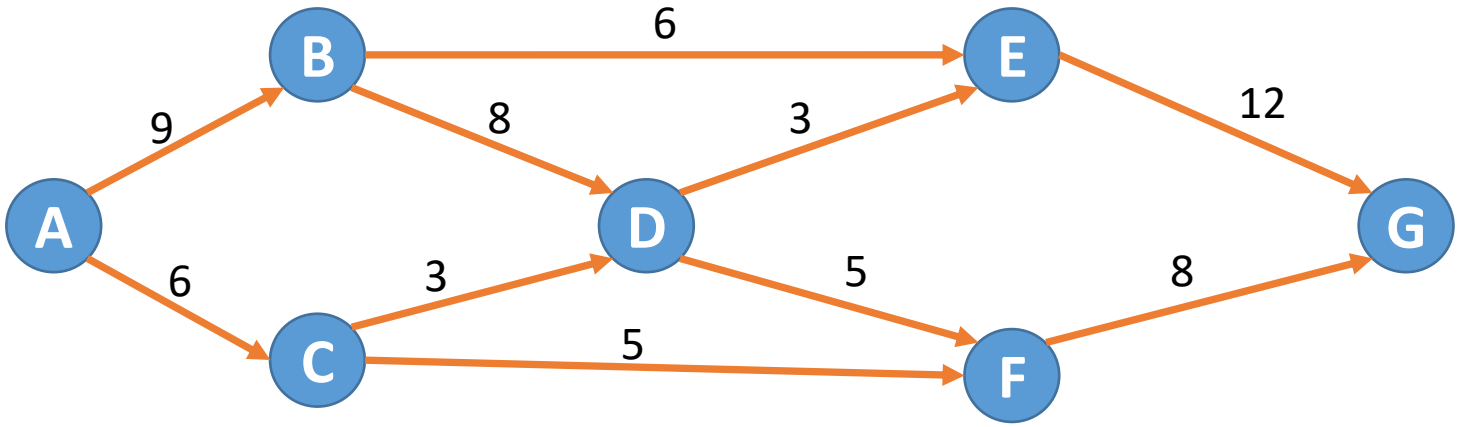
On repart des sommets atteints à partir de l'occurrence 5

- De F, on ne peut atteindre aucun sommet
- De I, on peut atteindre J (8 – 1)
- De J, on peut atteindre F (7 – 3)

k	A		B		C		D		E		F		G		H		I		J	
0	0		∞		∞		∞		∞		∞		∞		∞		∞		∞	
1			1	A	2	A	∞		∞		∞		∞		5	A	∞		∞	
2			0	C			4	B	3	B	6	H	∞				8	H	8	H
3							3	B	2	B	5	J	6	H					7	I
4									1	D	4	J	5	D			8	H		
5											4	J					8	H	7	I
6											4	J							7	I

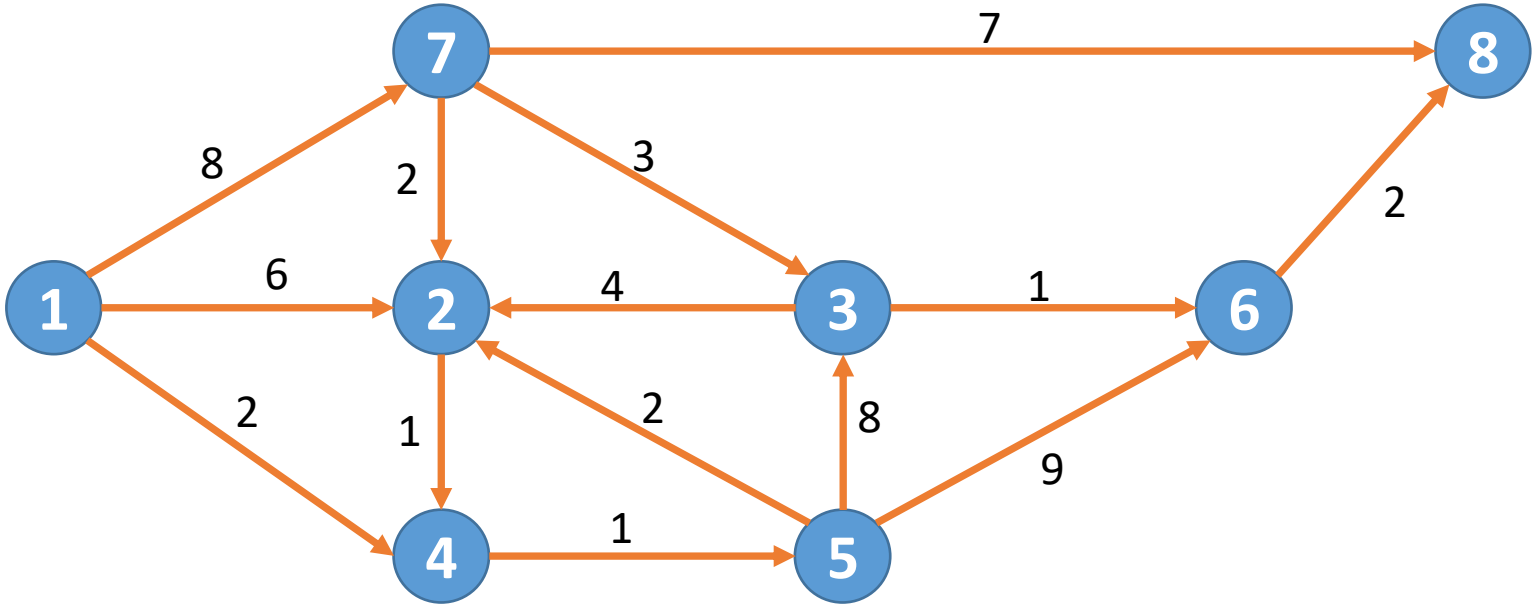
Exercice

Trouver le chemin le plus court entre les sommets A et G.



Exercice

Trouver le chemin le plus court entre les sommets 1 et 8.



Ordonnancement

Les problèmes d'ordonnancement sont apparus au départ dans la planification de grands projets

L'objectif est de gagner du temps sur leur réalisation

Les tâches du projet sont liées entre elles au travers de relations temporelles telles que:

- Une étape doit commencer à une date précise
- Un certain nombre de tâches doivent être terminées pour pouvoir en démarrer une autre
- Deux tâches ne peuvent être réalisées en même temps
- Chaque tâche nécessite une certaine quantité de main d'œuvre
- …

Ordonnancement

On cherche à établir un ordonnancement des tâches afin de minimiser le temps total de réalisation du projet

Nous allons uniquement nous concentrer sur des problèmes avec les contraintes:

- Une étape doit commencer à une date précise
- Un certain nombre de tâches doivent être terminées pour pouvoir en démarrer une autre

A partir de cette planification, nous allons constater que certaines tâches peuvent être modifiées sans impacter le projet, alors que d'autres sont considérées comme critiques et retardent le projet au moindre retard local

Ordonnancement

Il existe deux grandes méthodes pour résoudre des problèmes d'ordonnancement:

- La méthode américaine CPM (Critical Path Method) et sa variante PERT (Program Evaluation and Review Technique)
- La méthode française MPM (Méthode des potentiels)

Nous allons nous concentrer sur la méthode PERT

PERT

La méthode PERT s’organise en 6 étapes:

1. Préparer les tâches
2. Construire le réseau
3. Indiquer les dates au plus tôt
4. Indiquer les dates au plus tard
5. Calculer les marges des tâches
6. Déterminer le chemin critique

PERT (1 – Préparer les tâches)

La préparation des tâches consistent à lister toutes les tâches du projet, en faire une estimation de durée et établir les contraintes d'antériorité (les tâches qui doivent être terminées avant de pouvoir débuter une certaine tâche)

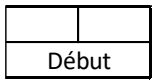
Tâches	Antériorité	Durée prévue
A	-	2
B	-	8
C	A	5
D	B	2
E	B	6
F	E	5
G	A - D	3

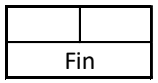
Dans nos exercices et / ou questions d'examen, cette tâche sera donnée (elle fait partie de l'énoncé)

PERT (2 – Construire le réseau)

A partir du tableau, on établit le réseau (graphe) en prenant

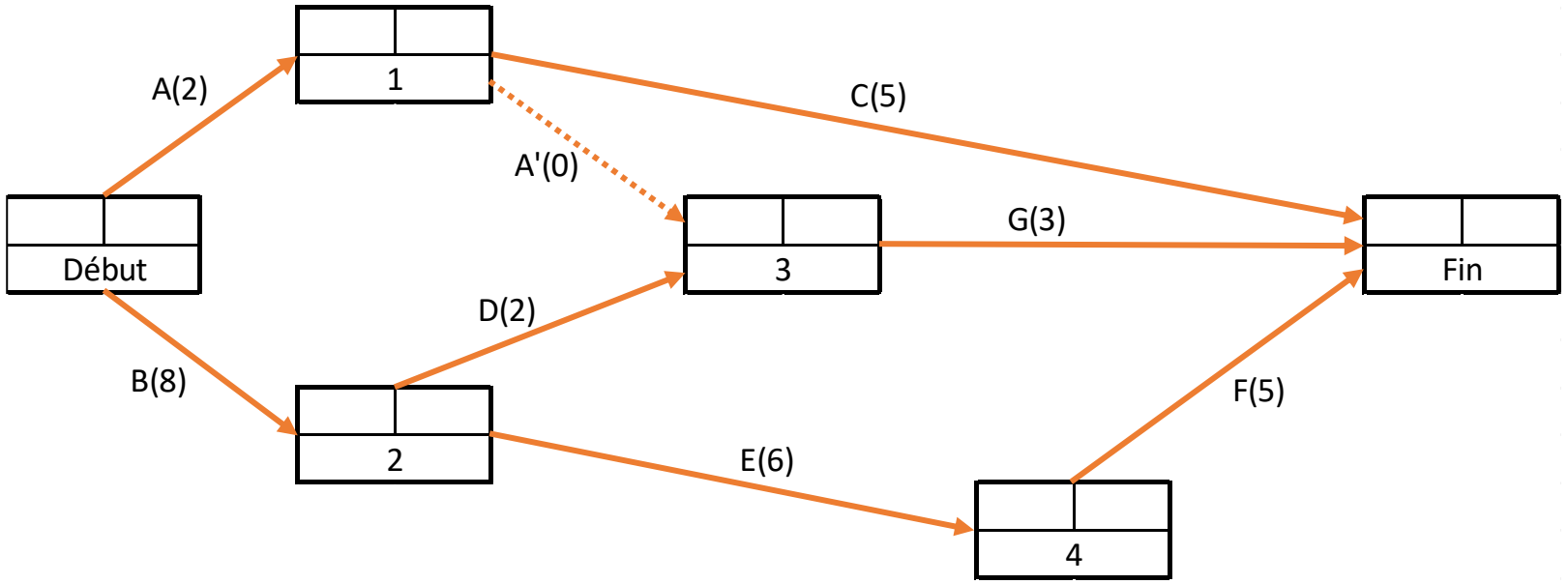
- Les sommets comme début de tâche
- Les arcs comme les tâches et leur durée prévue





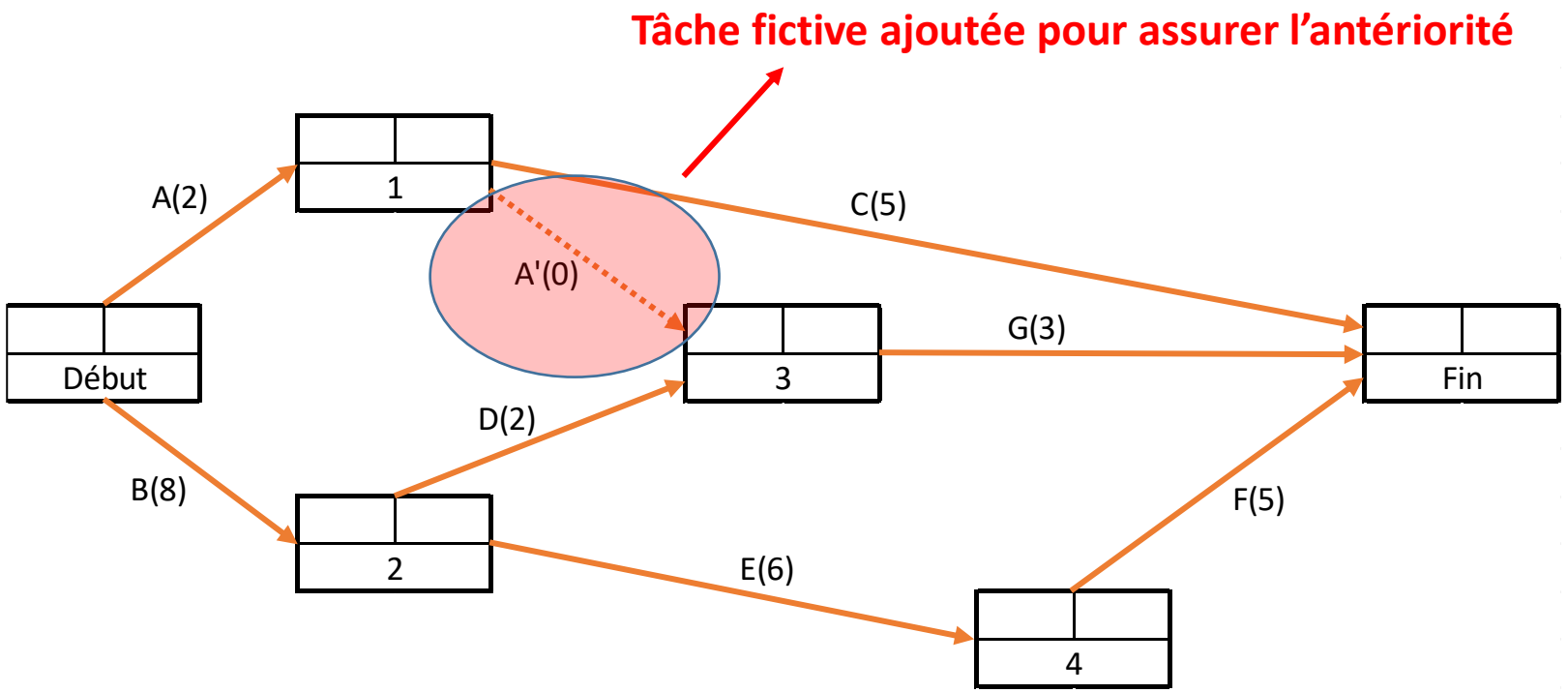
PERT (2 – Construire le réseau)

Tâches	Antériorité	Durée prévue
A	-	2
B	-	8
C	A	5
D	B	2
E	B	6
F	E	5
G	A - D	3



PERT (2 – Construire le réseau)

Tâches	Antériorité	Durée prévue
A	-	2
B	-	8
C	A	5
D	B	2
E	B	6
F	E	5
G	A - D	3



PERT (3 – Indiquer les dates au plus tôt)

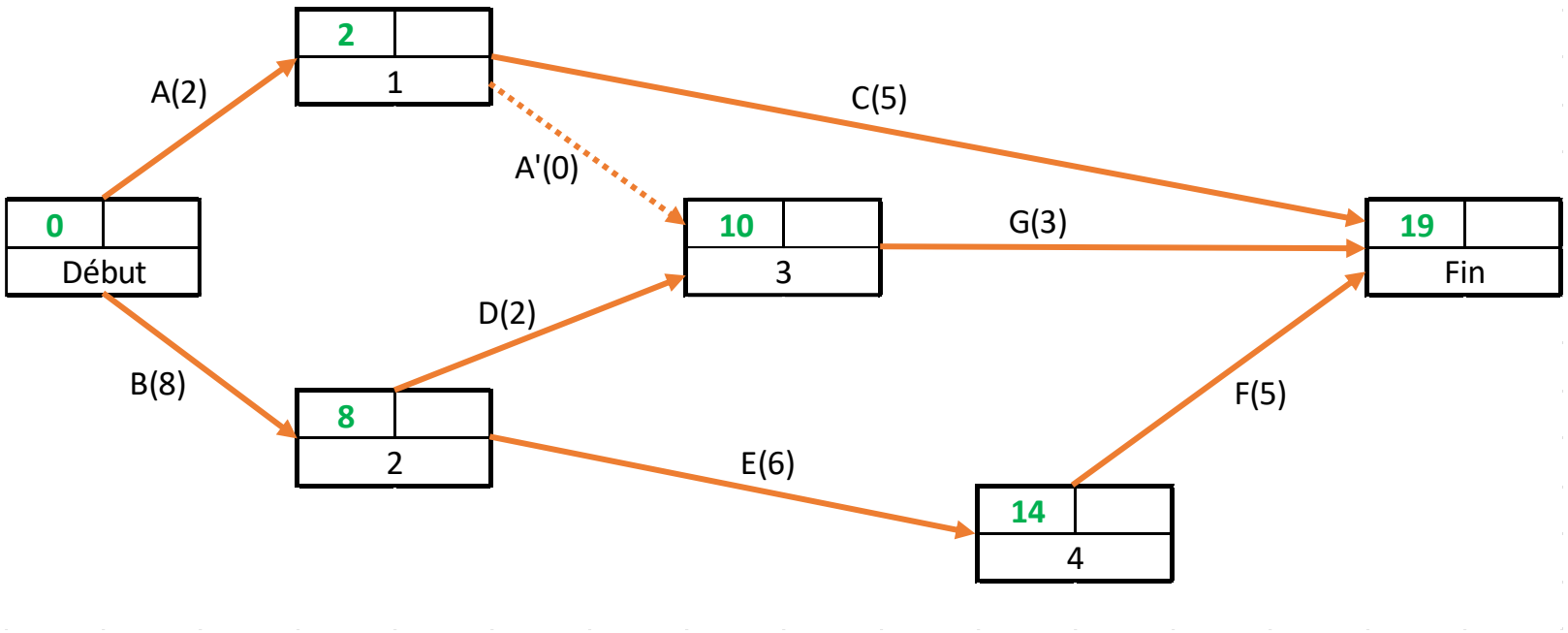
On débute au temps 0 et on ajoute pour chaque sommet la durée de la tâche qui aboutit au sommet

Lorsqu'un sommet est atteint par 2 tâches, on garde la durée la plus élevée

La date au plus tôt d'une tâche représente la date à laquelle elle peut débuter au plus tôt tenant compte des autres tâches du planning

- Exemple: La tâche G ne peut pas débuter avant 10 jours. La tâche A est terminée au bout de 2 jours, mais il faut 8 jours pour terminer la tâche B et 2 jours de plus pour terminer la tâche D

PERT (3 – Indiquer les dates au plus tôt)



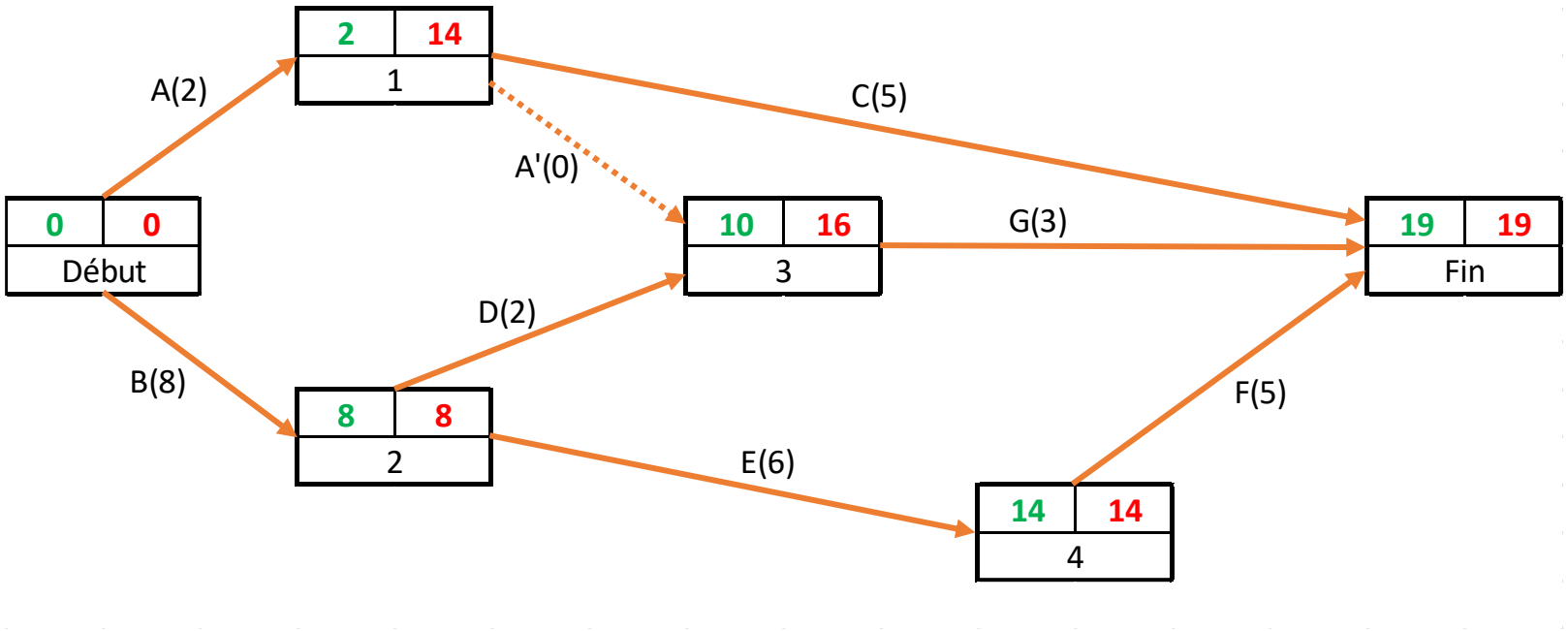
PERT (4 – Indiquer les dates au plus tard)

On débute au temps finale et on déduit pour chaque sommet la durée de la tâche qui débute au sommet

La date au plus tard d'une tâche représente la date à laquelle elle peut débuter au plus tard pour terminer le planning

- Exemple: La tâche C peut débuter au plus tard au jour 14. Comme elle dure 5 jours, elle ne peut pas débuter pus tard pour assurer la fin du planning au 19^{e} jour

PERT (4 – Indiquer les dates au plus tard)



PERT (5 – Calculer les marges des tâches)

La **marge totale** représente le retard que peut prendre la réalisation d'une tâche sans impacter la date de fin du projet

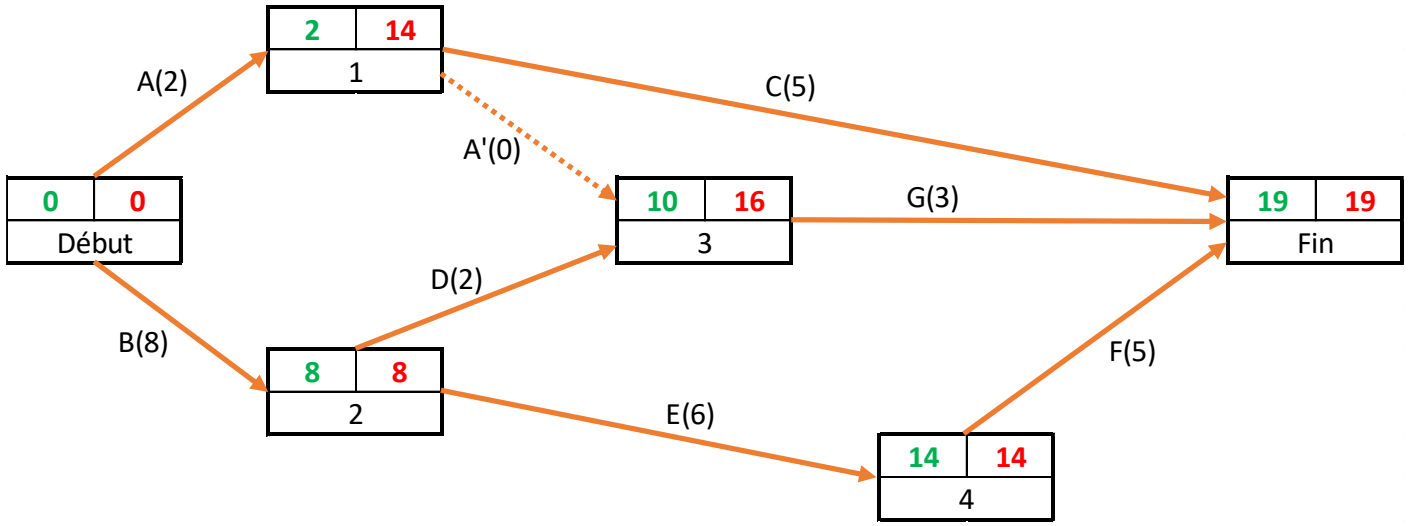
- Date au plus tard de l'étape suivante – Durée de la tâche – Date au plus tôt de l'étape précédente

La **marge libre** correspond au retard que peut prendre la réalisation d'une tâche sans impact sur la date au plus tôt des tâches suivantes

- Date au plus tôt de l'étape suivante – Durée de la tâche – Date au plus tôt de l'étape précédente

PERT (5 – Calculer les marges des tâches)

Tâches	Marge totale	Marge libre
A	12	0
B	0	0
C	12	12
D	6	0
E	0	0
F	0	0
G	6	6



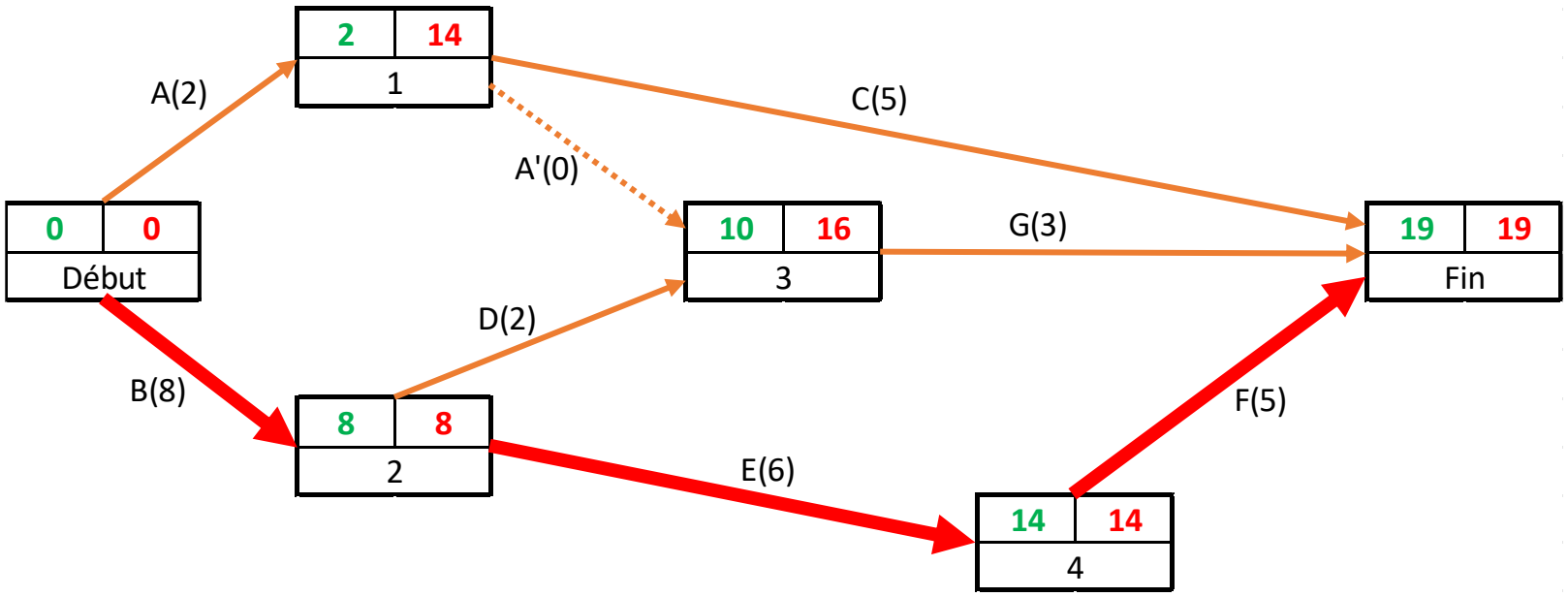
PERT (6 – Déterminer le chemin critique)

Une tâche critique est une tâche dont la marge totale est nulle

Le chemin critique est le chemin qui passe par les tâches critiques

C’est le chemin qui correspond au délai incompressible pour réaliser le projet

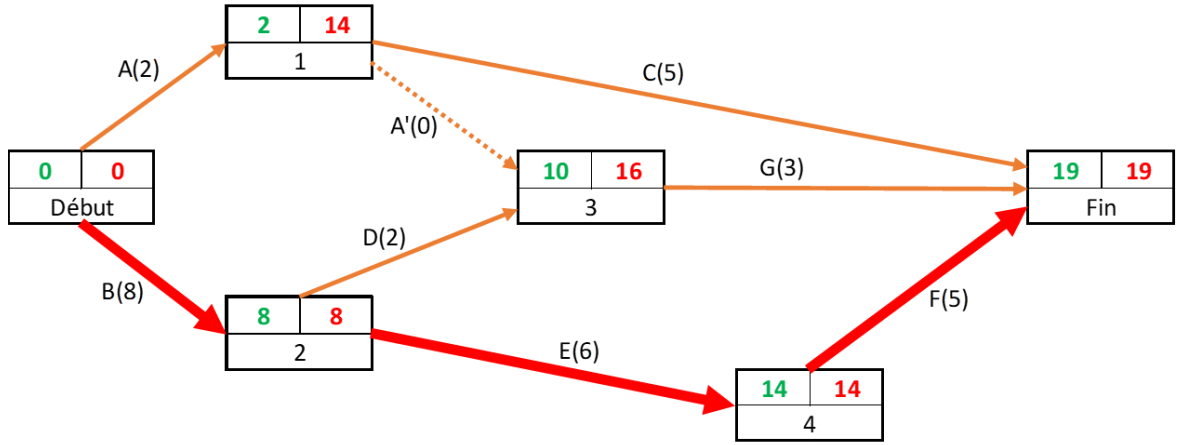
PERT (6 – Déterminer le chemin critique)



PERT

Tant que la tâche A est terminée pour le jour 14, le planning ne subira pas de retard

Si la tâche B est retardée, tout le projet est retardé



Exercice

Madame Legrand doit gérer le projet de construction d'une maison. Elle a déterminé les différentes tâches à accomplir, leur durée et les relations d'antériorité.

Déterminer le graphe PERT

Déterminer le chemin critique

Déterminer la durée minimale du projet

Tâches	Antériorité	Durée prévue
A	-	3
B	A	9
C	B	3
D	-	1
E	D	1
F	B	3
G	F	6
H	C - E - G	1

Exercice

Une papeterie dispose d'un groupe électrogène qui est utilisé pour suppléer la fourniture énergétique en cas de panne du réseau. Tous les ans, ce groupe électrogène doit être révisé. Le tableau suivant récapitule l'ensemble des tâches nécessaires à cette révision

Tâche	Description	Durée	Antériorité
A	Préparation du moteur	4	
B	Démointage et nettoyage du turbocompresseur	8	
C	Nettoyage de l'armoire électrique	1	
D	Vérification des contacteurs et du serrage des bornes	1	C
E	Révision de la pompe à injection	6	A
F	Changer les injecteurs	3	A
G	Remontage et contrôle du turbocmpresseur	5	B
H	Contrôle de l'injection	3	E - F - G
I	Vérification des différentiels	1	D
J	Contrôle de l'alternateur	2	I
K	Contrôle de l'avance	2	H
L	Contrôle du groupe électrogène	5	J - K

Exercice

Déterminer le graphe PERT

Déterminer le chemin critique

Déterminer la durée minimale du projet

Tâche	Description	Durée	Antériorité
A	Préparation du moteur	4	
B	Démointage et nettoyage du turbocompresseur	8	
C	Nettoyage de l'armoire électrique	1	
D	Vérification des contacteurs et du serrage des bornes	1	C
E	Révision de la pompe à injection	6	A
F	Changer les injecteurs	3	A
G	Remontage et contrôle du turbocmpresseur	5	B
H	Contrôle de l'injection	3	E - F - G
I	Vérification des différentiels	1	D
J	Contrôle de l'alternateur	2	I
K	Contrôle de l'avance	2	H
L	Contrôle du groupe électrogène	5	J - K

Séparation et évaluation progressive

La procédure de séparation et évaluation progressive (Branch and Bound) est une mise en application du principe « Divide ut Regnes »

C’est une procédure d’optimisation.

Le principe est de diviser l’espace de solutions en sous-ensembles de plus en plus petits

Séparation et évaluation progressive

Un marcheur dispose de 5 objets possibles à prendre avec lui pour partir en randonnée. Pour porter ces objets, il doit les mettre dans son sac à dos qui a une capacité de 6 l pour une masse maximale de 4 kg. Chaque objet apporte une satisfaction au marcheur, notée sur 10.

Quels sont les objets que le marcheur doit prendre avec lui en respectant les contraintes imposées par les caractéristiques de son sac à dos et en maximisant sa satisfaction?

Séparation et évaluation progressive

Le tableau suivant donne les caractéristiques de chaque objet:

Objets	Masse (kg)	Volume (l)	Satisfaction
Pull	1	2	6
K-way	0,5	1	4
Eau	2	2	10
Nourriture	0,5	1	8
Chaussure	2	2	4

Séparation et évaluation progressive

Il y a 2 façons de résoudre ce problème:

- On peut examiner chaque combinaison possible d'objets et prendre celle qui répond aux contraintes et maximise la satisfaction. 5 objets, signifie 2^5 combinaisons (32)
- On applique la procédure de séparation et évaluation progressive

Séparation et évaluation progressive

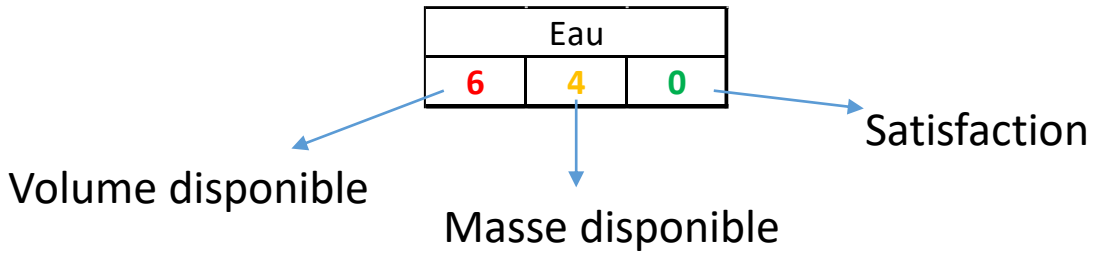
1ère étape: on classe les objets par satisfaction décroissante

Objets	Masse (kg)	Volume (l)	Satisfaction
Pull	1	2	6
K-way	0,5	1	4
Eau	2	2	10
Nourriture	0,5	1	8
Chaussure	2	2	4

Objets	Masse (kg)	Volume (l)	Satisfaction
Eau	2	2	10
Nourriture	0,5	1	8
Pull	1	2	6
K-way	0,5	1	4
Chaussure	2	2	4

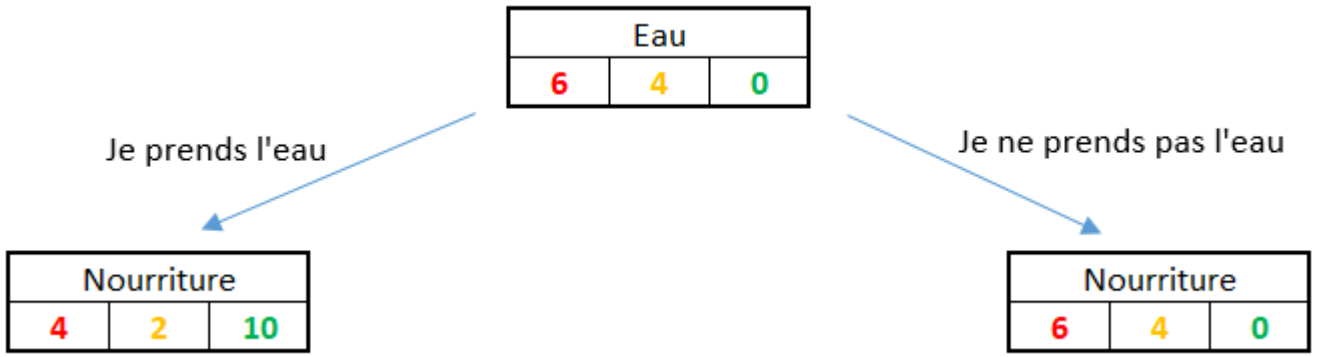
Séparation et évaluation progressive

2e étape: on construit un arbre de décision en partant du 1er objet (celui qui offre le plus de satisfaction.



Séparation et évaluation progressive

Pour chaque objet, on a le choix de le prendre ou de ne pas le prendre.



Et on continue tant qu'il y a des objets

Lorsque une branche ne permet plus de progresser car les contraintes ne sont pas respectées, on s'arrête

Séparation et évaluation progressive

Eau			Nourriture			Pull			K-Way			Chaussure			Total		
2	2	10	0,5	1	8	1	2	6	0,5	1	4	0,5	1	4			
1			1			1			1			1			4,5	7	32
1			1			1			1			0			4	6	28
1			1			1			0			1			4	6	28
1			1			1			0			0			3,5	5	24
1			1			0			1			1			3,5	5	26
1			1			0			1			0			3	4	22
1			1			0			0			1			3	4	22
1			1			0			0			0			2,5	3	18
1			0			1			1			1			4	6	24
1			0			1			1			0			3,5	5	20
1			0			1			0			1			3,5	5	20
1			0			1			0			0			3	4	16
1			0			0			1			1			3	4	18
1			0			0			1			0			2,5	3	14
1			0			0			0			1			2,5	3	14
1			0			0			0			0			2	2	10
0			1			1			1			1			2,5	5	22
0			1			1			1			0			2	4	18
0			1			1			0			1			2	4	18
0			1			1			0			0			1,5	3	14
0			1			0			1			1			1,5	3	16
0			1			0			1			0			1	2	12
0			1			0			0			1			1	2	12
0			1			0			0			0			0,5	1	8
0			0			1			1			1			2	4	14
0			0			1			1			0			1,5	3	10
0			0			1			0			1			1,5	3	10
0			0			1			0			0			1	2	6
0			0			0			1			1			1	2	8
0			0			0			1			0			0,5	1	4
0			0			0			0			1			0,5	1	4
0			0			0			0			0			0	0	0

Satisfaction maximale: 28

On prend

Eau – Nourriture - Pull – K-Way

ou

Eau – Nourriture – Pull - Chaussure

Exercice

Un voyageur prépare sa valise pour un séjour. La valise peut contenir un volume de 18 l et son poids ne peut pas dépasser 12 kg.

Objets	Masse (kg)	Volume (l)	Satisfaction
Pull	2	4	6
K-Way	1	3	5
Crème solaire	1	1	4
Eau	5	5	10
Nourriture	4	4	8
Chaussures	4	4	3

Déterminer les objets à emporter pour maximiser sa satisfaction

Algorithme de Little

Problème de voyageur de commerce

Un délégué commercial doit se rendre chez 4 médecins. Il part de chez lui et doit rentrer chez lui en fin de journée. Déterminer l'ordre de sa tournée pour qu'elle soit la plus économique possible sachant que les coûts de déplacement sont donnés par le tableau suivant

	A	B	C	D	E
A	-	10	12	8	14
B	10	-	9	8	7
C	12	9	-	15	6
D	8	8	15	-	11
E	14	7	6	11	-

Note: le tableau n'est pas nécessairement symétrique

Algorithme de Little

Pour chaque ligne, on supprime l'élément le plus petit

	A	B	C	D	E	
A	-	2	4	0	6	8
B	3	-	2	1	0	7
C	6	3	-	9	0	6
D	0	0	7	-	3	8
E	8	4	0	8	-	6

Algorithme de Little

Pour chaque colonne qui ne contient pas de 0, on supprime l'élément le plus petit

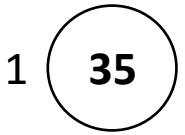
	A	B	C	D	E	
A	-	2	4	0	6	8
B	3	-	2	1	0	7
C	6	3	-	9	0	6
D	0	0	7	-	3	8
E	8	4	0	8	-	6
	0	0	0	0	0	

Algorithme de Little

On additionne les valeurs supprimées pour les colonnes et les lignes

8 + 7 + 6 + 8 + 6 = 35

Cela donne 1 premier nœud évalué à 35



Algorithme de Little

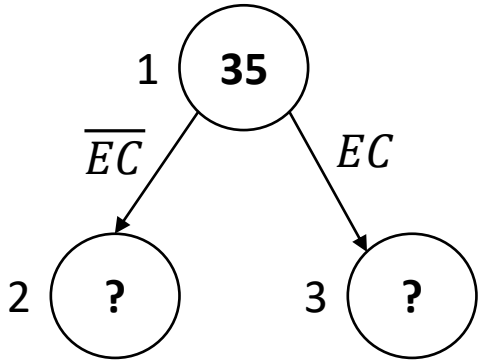
Pour chacune des cases annulées, on va calculer le « Regret » qui correspond à la somme du minimum de la ligne et minimum de la colonne de la case

	A	B	C	D	E	
A	-	2	4	0 (3)	6	8
B	3	-	2	1	0 (1)	7
C	6	3	-	9	0 (3)	6
D	0 (3)	0 (2)	7	-	3	8
E	8	4	0 (6)	8	-	6
	0	0	0	0	0	

Algorithme de Little

A partir de la nouvelle matrice, on va privilégier l'étude du trajet correspondant au « regret » le plus élevé: trajet E - C

En cas d'égalité, on fait un choix arbitraire

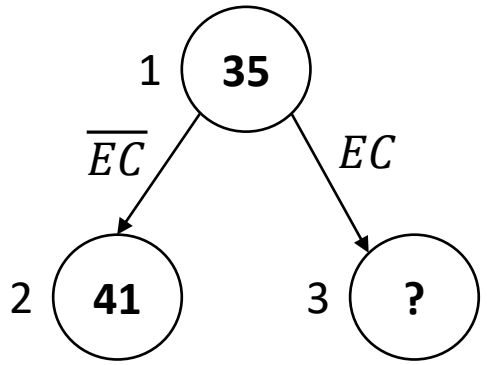


On va analyser le trajet en excluant le trajet E-C ou en incluant le trajet E-C

Algorithme de Little

L’évaluation du trajet excluant E-C est déjà valorisé et vaut le nœud précédent plus le « regret »: 35 + 6 = 41.

Cela représente le coût minimal de la tournée sans le trajet E - C



Algorithme de Little

Pour le trajet avec E – C, on recommence l'algorithme en repartant d'une nouvelle matrice de 1 degré de moins en supprimant la ligne correspondant à l'étape de départ (E) et la colonne de l'étape d'arrivée (C).

	A	B	C	D	E
A	-	2	4	0 (3)	6
B	3	-	2	1	0 (1)
C	6	3	-	9	0 (3)
D	0 (3)	0 (2)	7	-	3
E	8	4	0 (6)	8	-

	A	B	D	E
A	-	2	0	6
B	3	-	1	0
C	6	3	9	0
D	0	0	-	3

Algorithme de Little

Dans cette nouvelle matrice, on exclut les trajets qui entrainerait une boucle qui ne passe pas par toutes les étapes C – E

E – C puis C – E entraine qu'on revient au point de départ sans être passé par toutes les étapes.

	A	B	D	E
A	-	2	0	6
B	3	-	1	0
C	6	3	9	0
D	0	0	-	3

	A	B	D	E
A	-	2	0	6
B	3	-	1	0
C	6	3	9	-
D	0	0	-	3

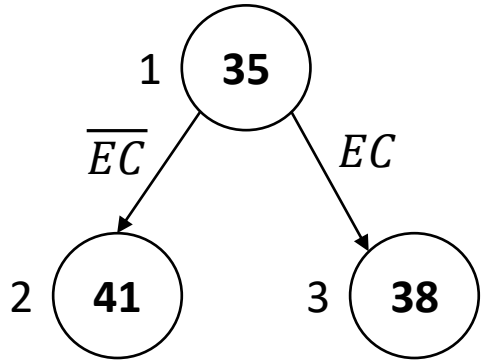
Algorithme de Little

On supprime les valeurs minimales de chaque ligne et colonne

	A	B	D	E	
A	-	2	0	6	0
B	3	-	1	0	0
C	3	0	6	-	3
D	0	0	-	3	0
	0	0	0	0	

Algorithme de Little

La somme des suppressions de ligne et des colonnes est ajoutée à la valeur du nœud 1 pour donner la valeur du nœud 3: 35 + 3 = 38



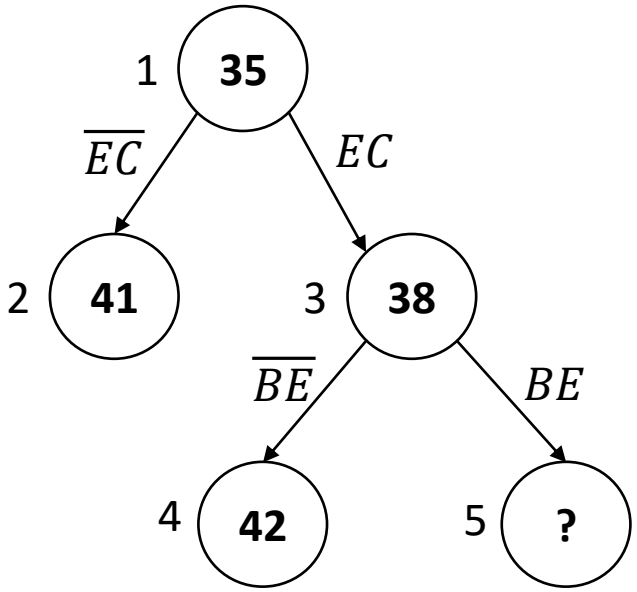
Algorithme de Little

On calcule les regrets

	A	B	D	E
A	-	2	0 (3)	6
B	3	-	1	0 (4)
C	3	0 (3)	6	-
D	0 (3)	0 (0)	-	3

Algorithme de Little

Le regret maximum (4) correspond au trajet B – E



La valeur du nœud 4 est donnée par la valeur du nœud précédent le regret maximum

Algorithme de Little

Réduction de la matrice

	A	B	D	E
A	-	2	0 (3)	6
B	3	-	1	0 (4)
C	3	0 (3)	6	-
D	0 (3)	0 (0)	-	3

	A	B	D
A	-	2	0
C	3	0	6
D	0	0	-

Algorithme de Little

Annulation du trajet C – B qui entrainerait une boucle incomplète

B – E E – C C – B

	A	B	D
A	-	2	0
C	3	0	6
D	0	0	-

	A	B	D
A	-	2	0
C	3	-	6
D	0	0	-

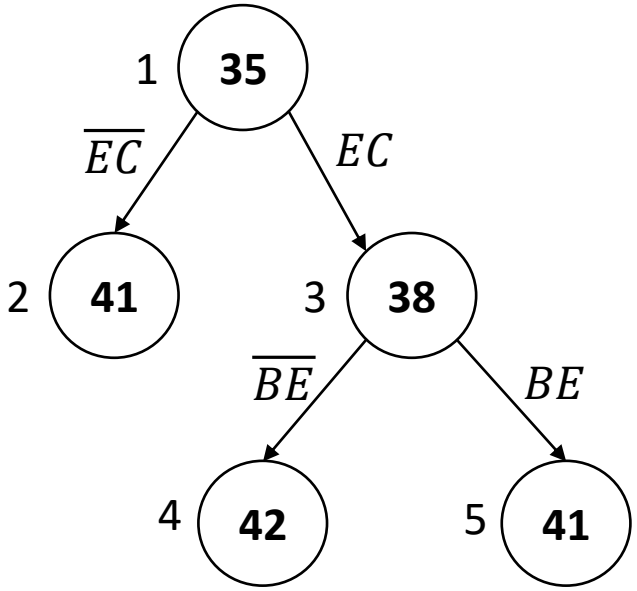
Algorithme de Little

On supprime les valeurs minimales de chaque ligne et colonne

	A	B	D	
A	-	2	0	0
C	0	-	3	3
D	0	0	-	0
	0	0	0	

Algorithme de Little

Valorisation du nœud 5



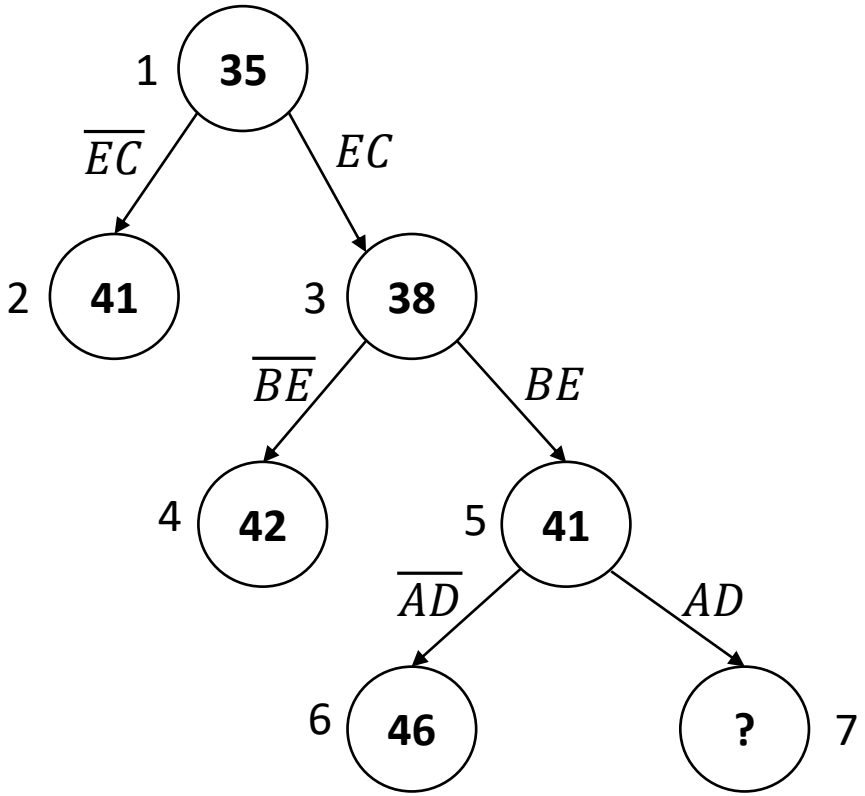
Algorithme de Little

Calcul des regrets

	A	B	D
A	-	2	0 (5)
C	0 (3)	-	3
D	0 (0)	0 (2)	-

Algorithme de Little

Le regret maximum (5) correspond au trajet A - D



Algorithme de Little

Nouvelle matrice et suppression des boucles incomplètes (D – A)

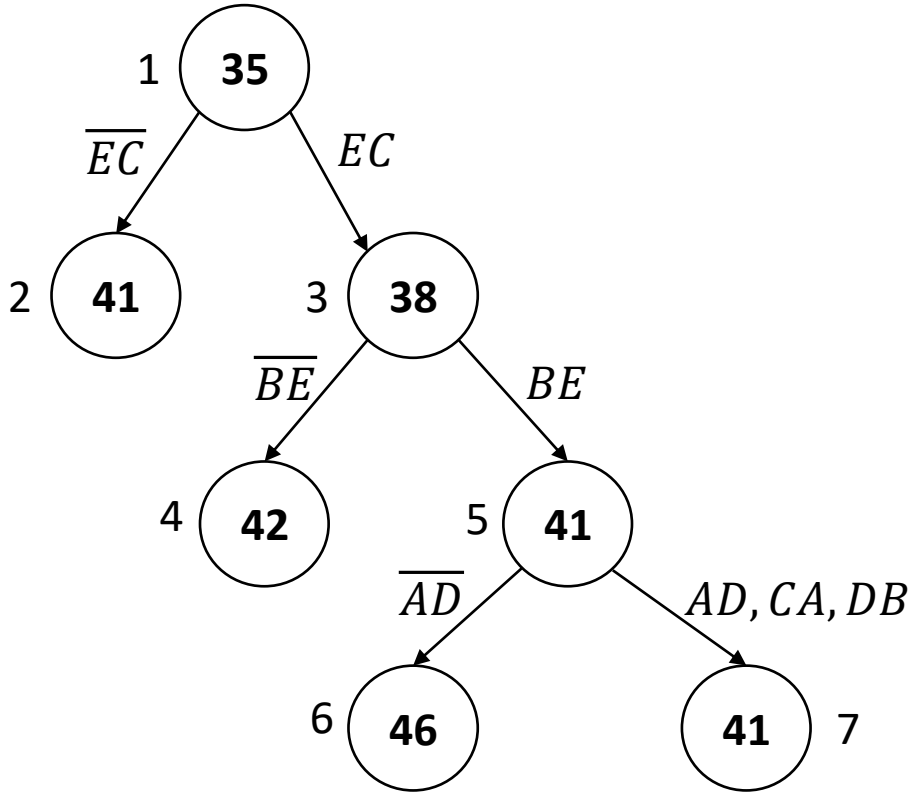
	A	B
C	0	-
D	-	0

Et on continue jusqu’à obtenir une matrice 1 X 1

Algorithme de Little

On remarque (avant d'obtenir la matrice 1 X 1) qu'il n'y a plus aucune réduction possible.

On intègre alors les trajets restants



Algorithme de Little

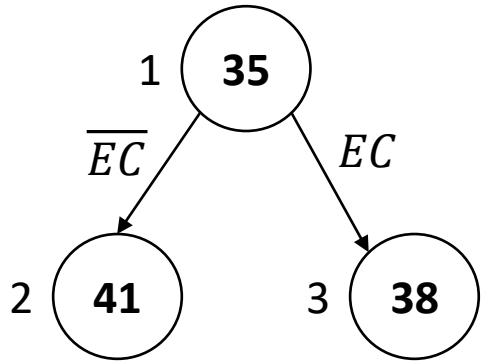
Nous avons donc un trajet C – A / A – D / D – B / B – E / E – C avec pour valeur 41

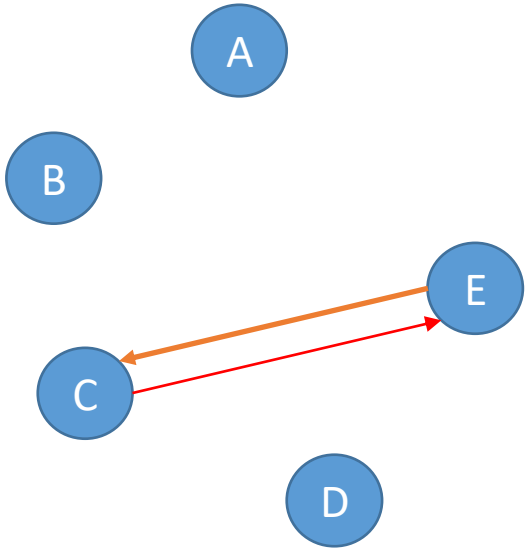
On revisite ensuite les nœuds 2, 4 et 6 pour vérifier si on ne peut pas améliorer le résultat.

Si valeur nœud revisité $\geq$ résultat -> On ne peut pas améliorer à partir de ce nœud

Algorithme de Little

Graphiquement



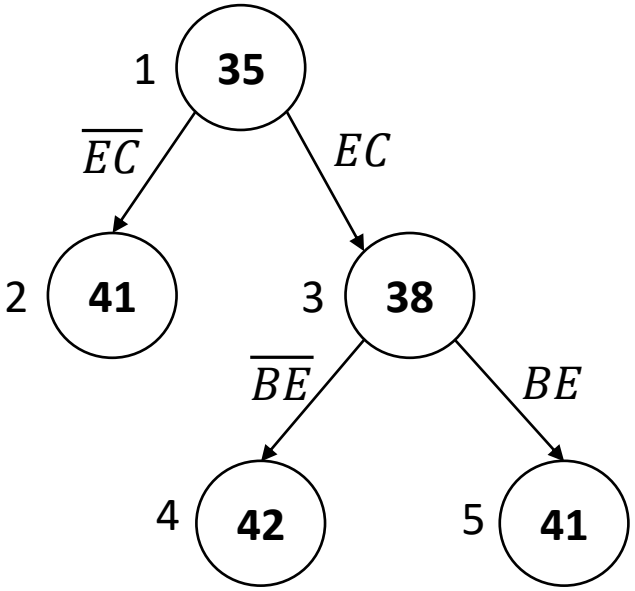


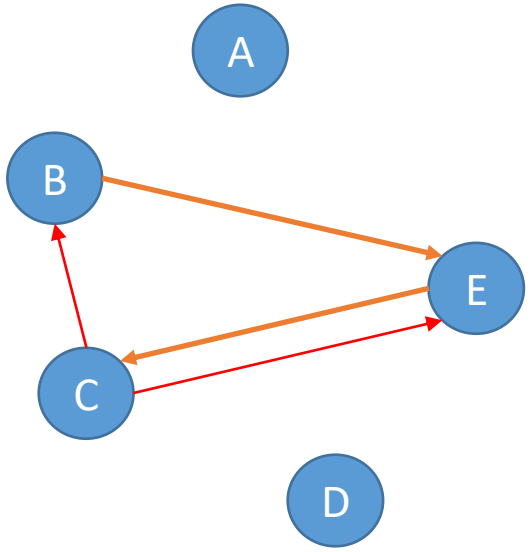
Trajet inclus

Trajet exclus

Algorithme de Little

Graphiquement



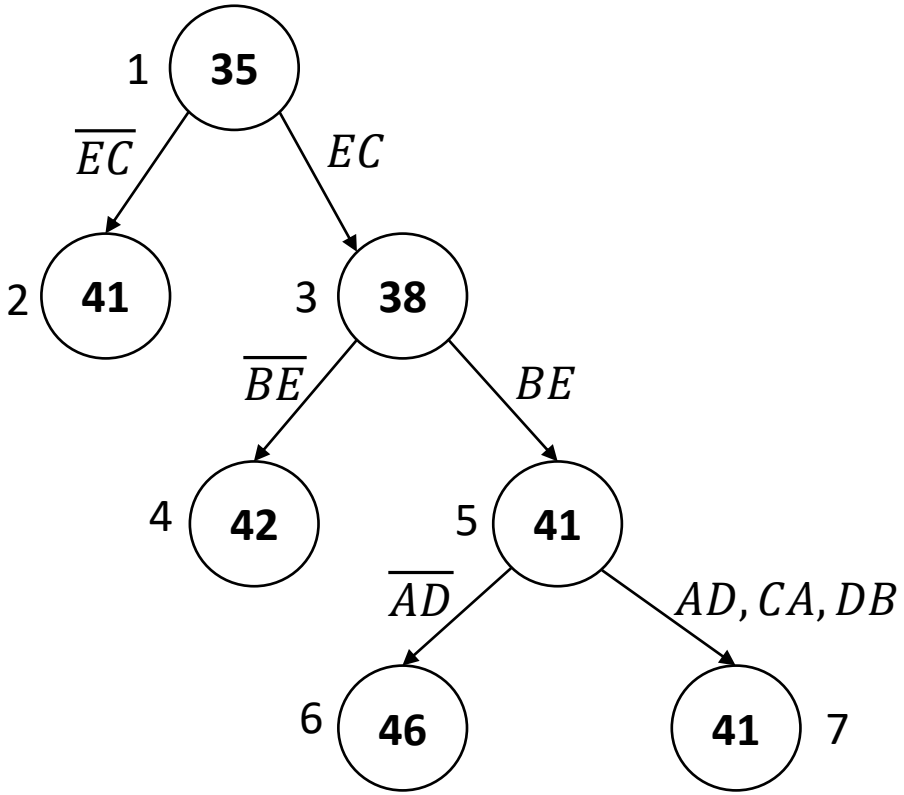


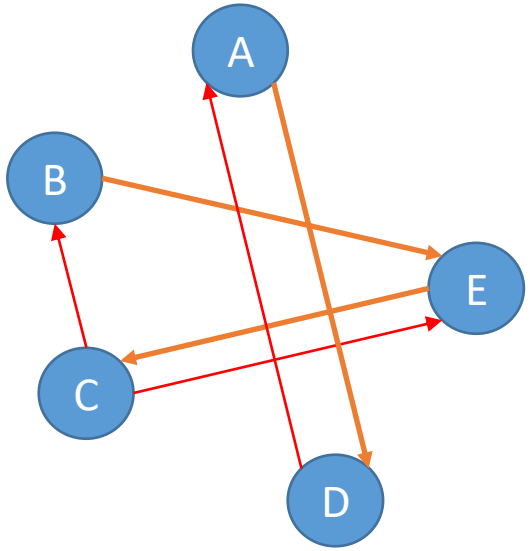
Trajet inclus

Trajet exclus

Algorithme de Little

Graphiquement



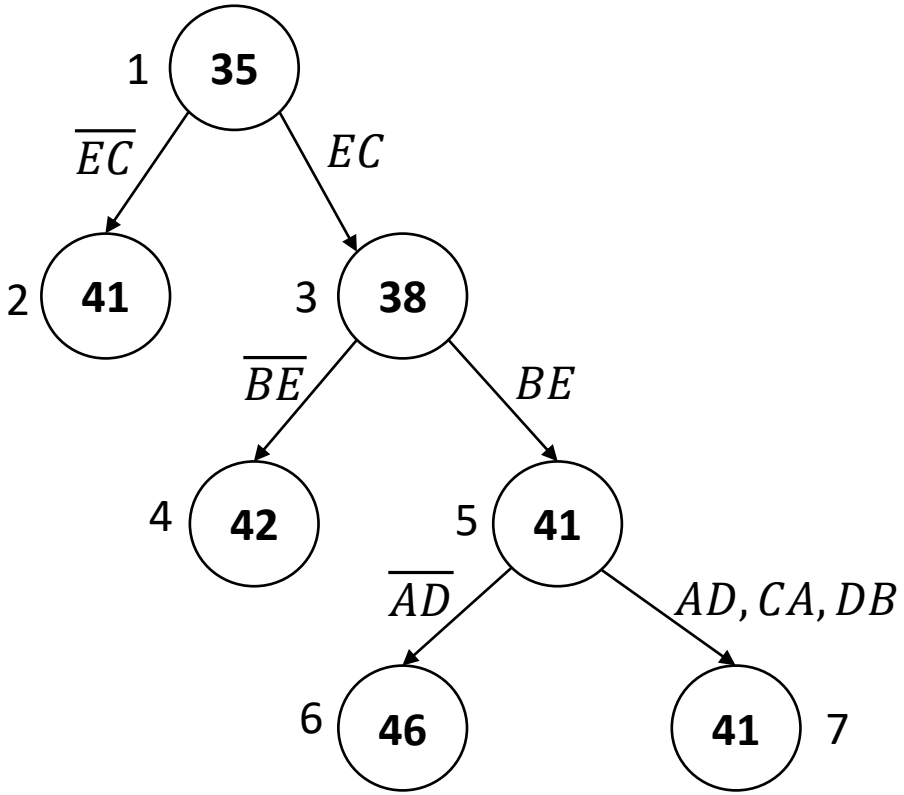


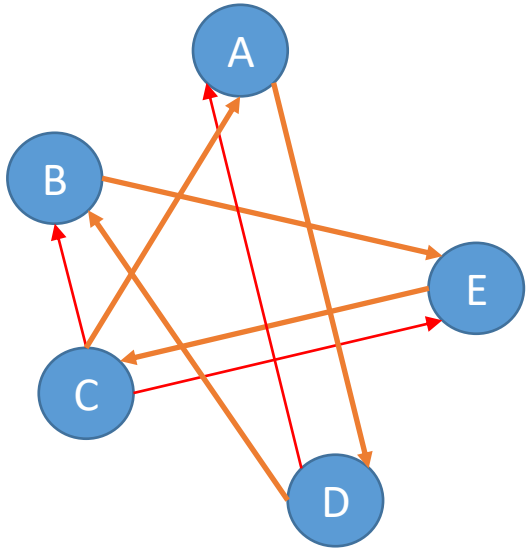
Trajet inclus

Trajet exclus

Algorithme de Little

Graphiquement





Trajet inclus

Trajet exclus

Algorithme de Little

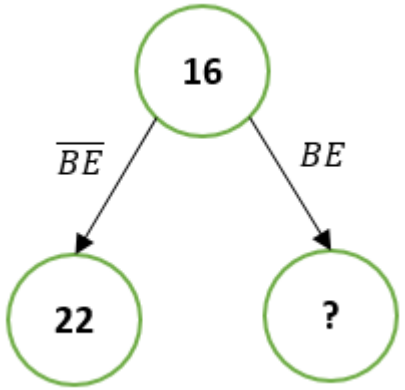
Que se passe-t-il si on doit revenir en arrière et examiner une branche « mise de côté » temporairement?

	A	B	C	D	E	F
A	-	1	7	3	14	2
B	3	-	6	9	1	24
C	6	14	-	3	7	3
D	2	3	5	-	9	11
E	15	7	11	2	-	4
F	20	5	13	4	18	-

Algorithme de Little

Etape 1

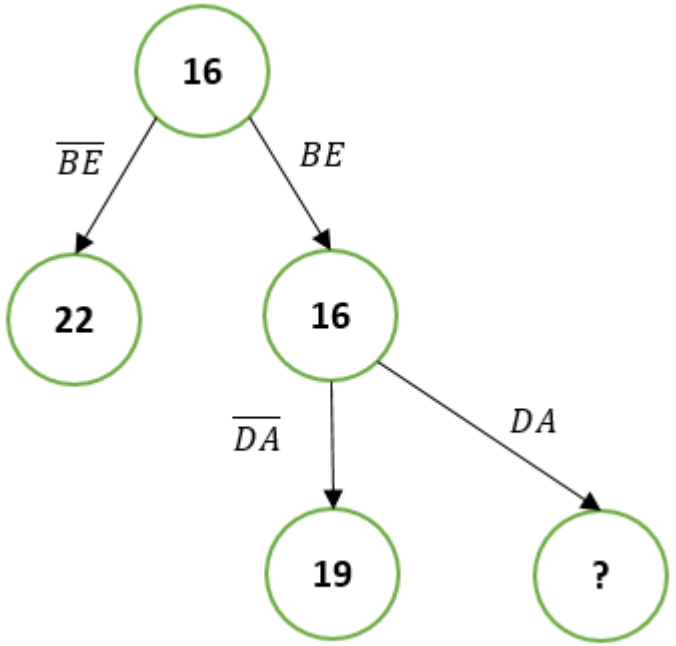
	A	B	C	D	E	F	
A	-	0 (2)	3	2	13	1	1
B	2	-	2	8	0 (6)	23	1
C	3	11	-	0 (0)	4	0 (1)	3
D	0 (3)	1	0 (3)	-	7	9	2
E	13	5	6	0 (2)	-	2	2
F	16	1	6	0 (1)	14	-	4
	0	0	3	0	0	0	**16**



Algorithme de Little

Etape 2

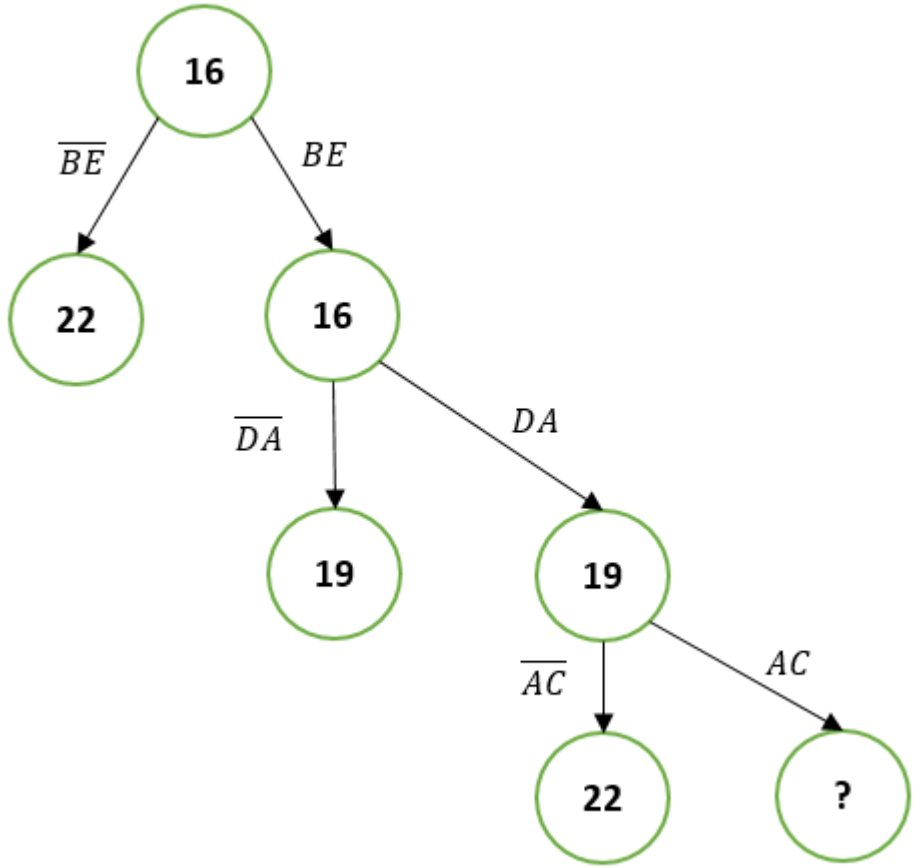
	A	B	C	D	F	
A	-	0 (2)	3	2	1	0
C	3	11	-	0 (0)	0 (1)	0
D	0 (3)	1	0 (3)	-	9	0
E	13	-	6	0 (2)	2	0
F	16	1	6	0 (1)	-	0
	0	0	0	0	0	**0**



Algorithme de Little

Etape 3

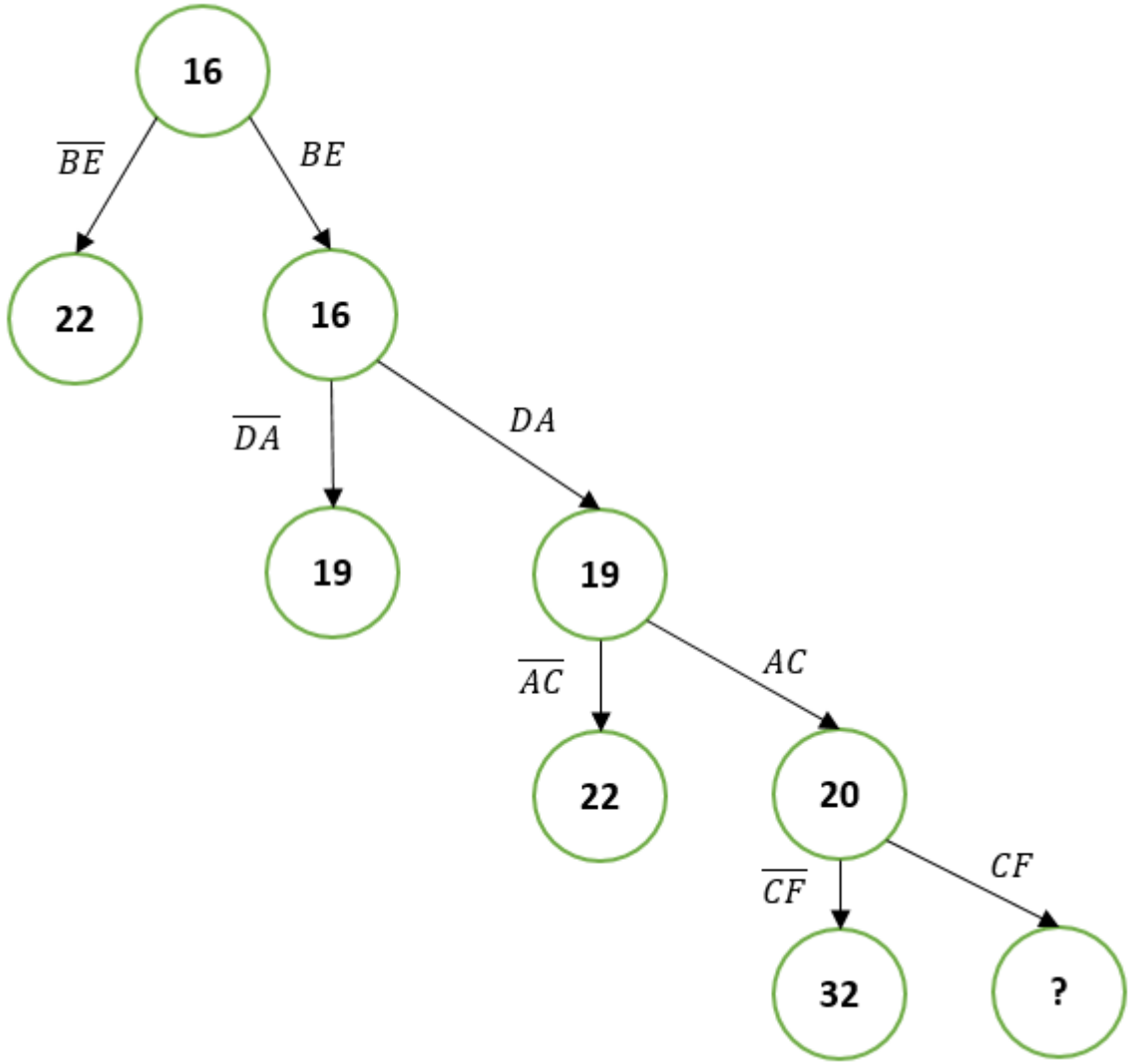
	B	C	D	F	
A	0 (1)	0 (3)	-	1	0
C	11	-	0 (0)	0 (1)	0
E	-	3	0 (2)	2	0
F	1	3	0 (1)	-	0
	0	3	0	0	**3**



Algorithme de Little

Etape 4

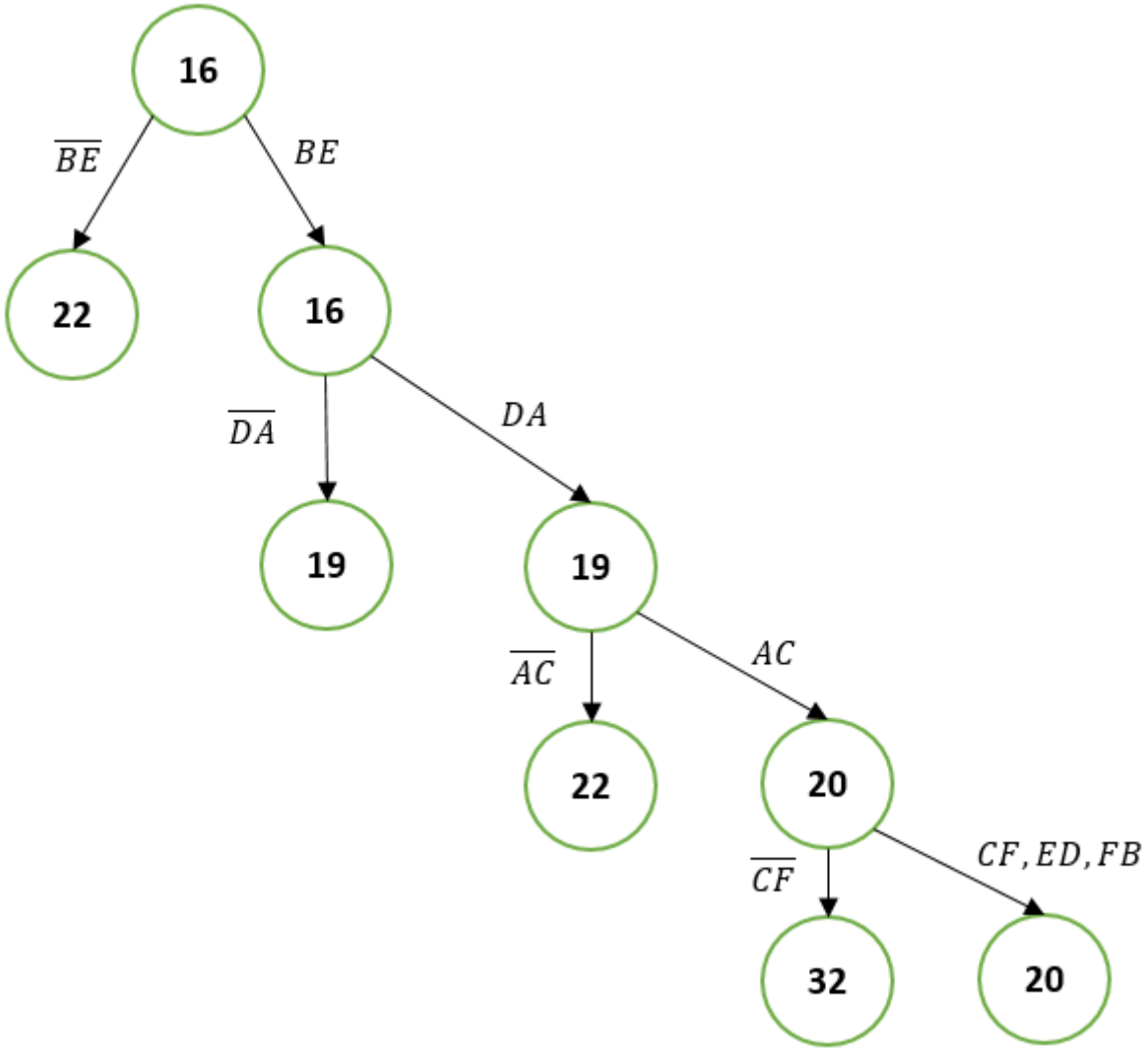
	B	D	F	
C	10	-	0 (12)	0
E	-	0 (2)	2	0
F	0 (10)	0 (0)	-	0
	1	0	0	**1**



Algorithme de Little

Etape 5

	B	D
E	-	0
F	0	-



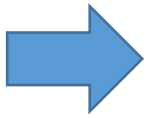
Algorithme de Little

Etape 6

La branche $\overline{DA}$ a un coût inférieur à 20, donc on repart de cette branche pour explorer le graphe et vérifier si il y a un chemin moins couteux

On repart donc de la matrice avant la suppression du chemin DA et on élimine ce chemin

	A	B	C	D	F
A	-	0	3	2	1
C	3	11	-	0	0
D	0	1	0	-	9
E	13	-	6	0	2
F	16	1	6	0	-

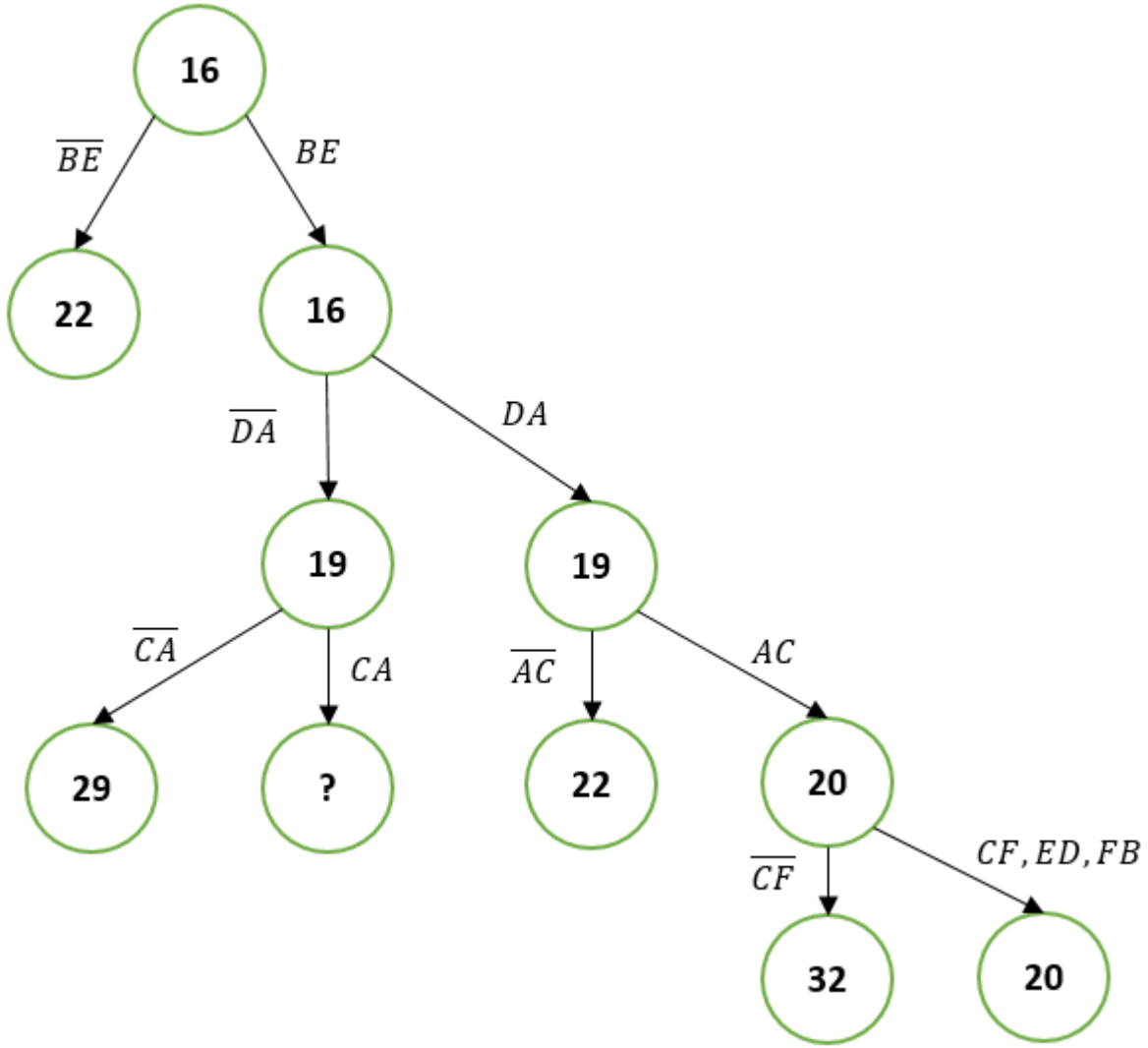


	A	B	C	D	F
A	-	0	3	2	1
C	3	11	-	0	0
D	-	1	0	-	9
E	13	-	6	0	2
F	16	1	6	0	-

Algorithme de Little

Etape 6

	A	B	C	D	F	
A	-	0 (2)	3	2	1	0
C	0 (10)	11	-	0 (0)	0 (1)	0
D	-	1	0 (4)	-	9	0
E	10	-	6	0 (2)	2	0
F	13	1	6	0 (1)	-	0
	3	0	0	0	0	**3**

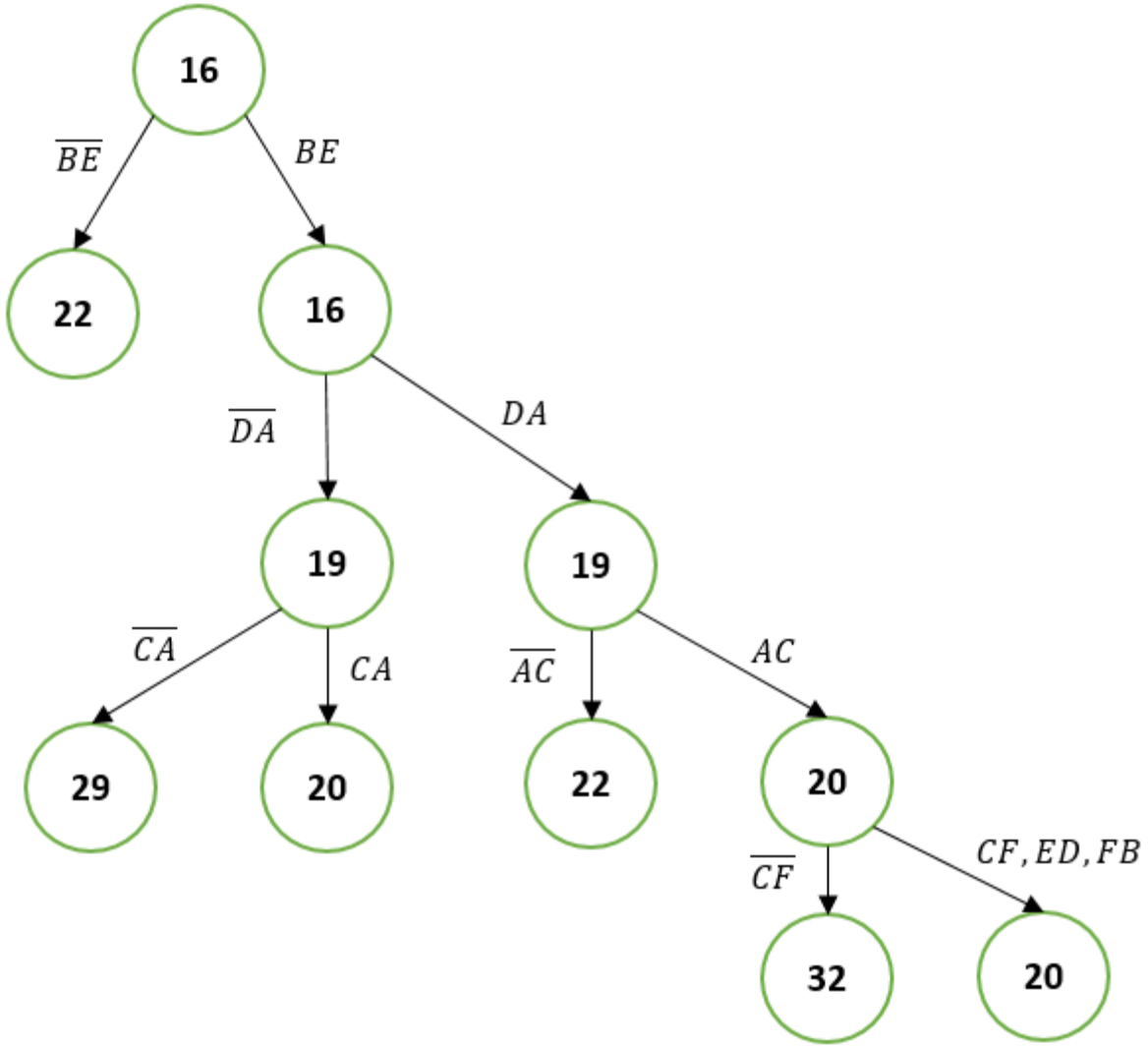


Algorithme de Little

Etape 7

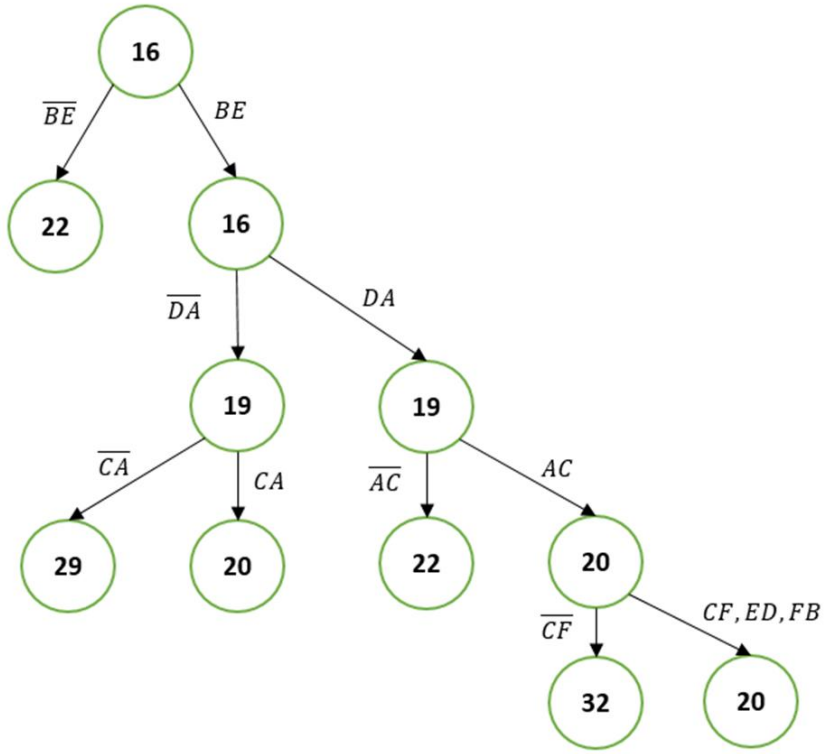
	B	C	D	F	
A	0 (2)	-	2	0 (1)	0
D	1	0 (4)	-	8	0
E	-	6	0 (1)	1	0
F	1	6	0 (1)	-	0
	0	0	0	1	**1**

La nouvelle Branche CA coûte au minimum 20. Hors notre chemin initial valait 20. Donc poursuivre $BE - \overline{DA} - CA$ ne va pas améliorer le coût. Au mieux on aura le même coût



Algorithme de Little

Donc le chemin à suivre est $BE - ED - DA - AC - CF - FB$



Exercice

Un voyageur doit visiter successivement 5 villes et finalement revenir à sa ville de départ. En sachant que les coût de passage d'une ville à une autre sont donnés par la matrice suivante:

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix}$$

Trouvez le chemin offrant le coût le moins élevé

Exercice

Un voyageur doit visiter successivement chacune des 6 villes suivantes: Bordeaux (B), Lyon (L), Nantes (N), Paris (P), Montpelier (M) et Dijon (D) et revenir à son point de départ en effectuant le chemin le plus court.

	B	L	N	P	M	D
B	-	780	320	580	480	660
L	780	-	700	460	300	200
N	320	700	-	380	820	630
P	580	460	380	-	750	310
M	480	300	820	750	-	500
D	660	200	630	310	500	-